High Saturation Complete Graph Approach for EC Point Decomposition and ECDL Problem

Nicolas T. Courtois

University College London, Computer Science, Room 6.18. Gower Street, WC1E 6BT, London, UK
n.courtois@ucl.ac.uk

Abstract. One of the key questions in contemporary applied cryptography is whether there exist an efficient algorithm for solving the discrete logarithm problem in elliptic curves. The primary approach for this problem is to try to solve a certain system of polynomial equations [39]. Current attempts try to solve them directly with existing software tools which does not work well due to their very loosely connected topology [38, 35] and illusory reliance on degree falls [31]. A deeper reflection on what makes systems of algebraic equations efficiently solvable is missing. In this paper we propose a new approach for solving this type of polynomial systems which is radically different than current approaches, cf. [38, 35]. We carefully engineer systems of equations with excessively dense topology obtained from a complete clique/biclique graphs and hypergraphs and unique special characteristics. We construct a sequence of systems of equations with a parameter K and argue that **asymptotically** when K grows the system of equations achieves a high level of saturation with $\lim_{K\to\infty} F/T = 1$ which allows to reduce the "regularity degree" and makes that polynomial equations over finite fields may become efficiently solvable.

Key Words: cryptanalysis, finite fields, elliptic curves. ECDL problem, Semaev polynomials, block ciphers, NP-hard problems, MQ problem, phase transitions, XL, Gröbner bases, ElimLin

Table of Contents

1 A Short Explanation for Beginners

In order to explain in just a few words what we do in this paper, and what is all the fuss about, consider the following ultra simplified example. The dominant paradigm for solving the ECDL problem is to solve a certain system of polynomial equations using so called Semaev polynomials. We consider an ultra-simplified version of this problem with just two points. The attacker wants to solve the following problem, essentially nothing really more complex that:

$$P1 + P2 = Q$$

where $P1/P2$ satisfy some extra constraints, not every $P1$ is permitted. The standard method to do this which dominated in the literature is use Semaev polynomials, [39]. We bring a standard piece of software which converts our EC equations to polynomials. Then we bring another standard piece of software which **maybe** solves the equations with a so called Gröbner basis computation.

This wishful thinking approach has been a miserable failure [27, 31, 35]. One crucial problem is that the complexity of this process depends on a certain integer called "regularity degree" cf. [2] and would be solved in polynomial time if this integer was a small constant independent on input size n. Actually exactly the contrary happens, cf. [31], we hit the wall, for example some of the most recent attacks of this form have running times growing exponentially cf. [35].

The regularity degree however is NOT an absolute limitation in algebraic cryptanalysis, cf. [9]. The attacker needs to work on **reducing** this degree. And this is in fact very easy, not harder than in [9]. For example consider the following problem:

$$\begin{cases} P1 + P2 = Q \\ P1 + D = P1' \\ P2 + E = P2' \\ P1' + P2' = (Q + D + E) \end{cases}$$

Here Q, D, E and $Q + D + E$ are constants. Now we run our two pieces of software again. Oops, **the degree of regularity has decreased.** Maybe, or so we claim[1]. As simple as that. And we can iterate this simple expansion process at will. We do no longer have an obscure "plug and pray" approach in which some mysterious things such as degree falls might happen, but we **explicitly engineer** a new system of equations which is easier to solve. Specialists will say: we applied a "Linear EC Code" expansion to reduce the "regularity degree".

We could stop here, claim that we have broken the EC discrete log and let the reader figure out all the little details. However there is a considerable amount of little details hence we wrote a longer paper. Moreover there is a very serious flaw in all the above methodology which basically **cannot work** as presented. Further reading will allow the reader to discover what the problem is.

[1] This is essentially because we have added 2 new variables and 3 new equations. Our system of polynomial equations will now be more **overdefined**, cf. [6]. And this could be sufficient enough to make it easier to solve cf. [6, 7, 2, 1, 18, 5, 17, 8, 9] while EC group law has produced "random-like" new constant $Q + D + E$ like new "non-trivial" information coming from an oracle.

Part I

Introduction

2 Introduction - Index Calculus and EC Crypto

At Crypto 1985 conference Victor Miller [34] have suggested that:

> It is extremely unlikely that an index calculus attack
> on the elliptic curve method will ever be able to work.
> V. Miller: "Use of elliptic curves in cryptography", Crypto'85, [34].

For the last 30 years this sort of impossibility, or a belief in impossibility, has been the basis on which the security of elliptic curve cryptosystems was based. More recently a number of papers in cryptography have attempted to nevertheless construct an index calculus for the EC discrete log problem. The first attempt in this direction is the 2004 paper by Semaev cf. [39]. The suggested method is to try to split a given point on an elliptic curve into a sum of several points lying in a certain sub-space. This is expected to be achieved through solving a system of polynomial equations derived from so called summation polynomials, cf. [39]. More recently several authors have suggested that this sort of method could eventually lead to sub-exponential algorithms at least for the binary elliptic curves [38, 36, 26, 29, 27].

All current attempts operate by writing some systems of equations and trying to try to solve them directly (write and solve) with existing software tools such as Gröbner basis algorithms or SAT solvers. However not enough attention is paid to the questions of what makes systems of algebraic equations efficiently solvable.

2.1 Current Algebraic Approaches

It is easy to see that all current attempts are deeply flawed on two accounts.

First of all, they do not work well due to their very loosely connected topology [38, 35], with a risk that Gröbner basis algorithms will never succeed to create equations which relate various variables in the system, and this probably explains the excessively poor scalability of results in all numerical examples published to date cf. for example the simulations in [35] which exhibit very fast exponential-like growth in the running times.

Secondly, we have this naive assumption that when degree falls occur, a certain system of equations might collapse and will be solvable, without too much additional effort. This sort of assumption definitely works in some cases and the system of equations is solved completely [4, 33, 31] but sometimes the process may run out of steam after making a lot of progress [8, 9, 5, 37]. Moreover this assumption is simply wrong and it is simply not true in general, cf. [31].

2.2 Binary vs Mod P Curves

Interestingly, in the original paper from 2004, a variant of this method was also proposed for the "ordinary" elliptic curves modulo p, which is the main form of elliptic curve cryptography which is used in the real-life systems, for example in TLS, SSH, bitcoin etc.

Even though there is no evidence that any of the attacks proposed for binary curves actually works [27, 31], recently researchers have tried to propose some innovations which make attacks for mod P curves more plausible.

A very recent paper by Kosters Petit and Messeng [March 2016] proposes a new method to encode the problem of DL in elliptic curves designed specifically to work well for curves such that $P-1$ has many small factors which is the case for standard NIST curves for example P-224, cf. [35]. Another method based on EC isogenies designed to work for arbitrary P is also proposed and tentatively implemented.

2.3 Poor Results

Both methods suffer from a serious problem which is the same as in all works in this space published in the last few years: there is no evidence that the method works at all except on very small examples[2]. In fact recent works very clearly explain that [first] degree falls do NOT guarantee that system of equations becomes efficiently solvable [31] and all recent papers on this topic require this assumption to produce an attack on ECDL problem. In general it is quite interesting to see that all these works speculate about how to solve ECDL problem using a point-splitting step, however no point-splitting of any sort was ever demonstrated in practice for any realistic elliptic curve. On the contrary, there is a very substantial qualitative and quantitative gap between what kind of toy attacks can be demonstrated in practice with current tools and what would be needed to break any even remotely realistic form or elliptic curve cryptography.

2.4 What Is New

In this paper we propose a different approach, unlike any approach yet proposed in the literature. We postulate that in contrast with systems of equations over $GF(2)$ already massively exploited in cryptanalysis [6, 7, 2, 1, 4, 18, 5, 17], one needs to work A LOT harder in order to produce a system of equations over a larger finite field which can be solved. Our approach is to generate a sequence of systems of equations which are increasingly more and more overdefined up to a point of "saturation" where they could have better[3] chances of becoming efficiently solvable.

The crucial question we are going to study is first to demonstrate the existence of new polynomial equations which are new and NOT contained in the ideal or Gröbner basis generated by the Semaev polynomials in previous approaches and then show HOW they can be used to construct certain **very highly overdefined systems** of equations with unique strong characteristics of high connectivity and density with interesting asymptotic properties.

[2] In fact from simulations in Sec. 4.2. of [35] we can deduce that the method does not work, as the time complexity seems to grow faster than 2^n.

[3] At least better than in any previous approach. It is possible to see that even low degree and very highly saturated equations over both $GF(2^n)$ and $GF(p)$ fields can still be very hard to solve, at least on contrived examples, for example when they have many solutions, and handling these multiple solutions is a major difficulty (here and elsewhere).

3 The Road Map

Before we design our new attack which allows to split a point on the elliptic curve, we are going to explain what kind of characteristics we would like our attack to have. Our main strategy can be described in terms of what we don't want to have and a list of goals for an alternative approach.

1. We do not want to write a certain system of equations and hope it is efficiently solvable which is a frequent and major difficulty in algebraic cryptanalysis [39, 29, 38, 35, 21, 19].
   We want to engineer/contruct systems of equations which are made to be efficiently solvable.
2. We don't want to deal with a question whether a certain system of equations can be solved by Gröbner bases F4, XL or FXL after expansion to a certain degree. This degree is typically called a "regularity degree", cf. [2, 1]. We don't want to worry how quickly this degree will grow when the parameter n grows[4].
3. We do not want obscurity, we want clarity.
4. We don't want degree falls just to happen, maybe accidentally.
   We want to explicitly design, generate, enumerate/count and analyse the degree falls.
5. We would like to construct a system of equations where equations are quadratic or cubic, and they remain quadratic at every stage, and they are solved at degree 2 or 3. Or that they are eventually solved at degree $2 + \varepsilon$ where ε is small.

This sort of strategy is not new. For example it is very clearly proposed in slides 82-86 of [17]: avoid expansion stage[5]. This approach can be seen as one of the several possible so called "Fast" algebraic attacks strategies, cf. slide 84 in [17] and slide 121 in [19].

3.1 Additional Postulates

We further postulate that:

1. Experience shows that sometimes we can work with systems of equations which are larger than initially yet they can be overall easier to solve, cf. for example [8, 5, 17].
   One major way to achieve this is to add new variables, which is major paradigm shift compared to methods which are usually used in this space, and a major general direction of work, cf. Section 3.4. In particular, a major family of techniques in this space are the [linear] **EC Codes**, cf. Section 3.7 below.

[4] Many previous work have naively assumed that this degree is fixed, cf. for example [29, 30, 38], while in fact it increases with n.

[5] Such "expansion" stage is really the essence of all XL, mutant XL, F4, and other Gröbner basis methods, cf. slides 72-75 in [19] and [1, 3, 6, 2]. .

2. Working at degree 2 makes it potentially a lot easier to understand if our system of equations is efficiently solvable or not.
   Either it generates enough linear equations with ElimLin [5] or maybe it can be solved by some more advanced method[6], which operates without increasing the degree, or maybe it is hard to solve.
3. Our complexity claims should be based on a number of facts which highly regular, highly predictable and possible to validate experimentally for a wide range of parameters.
4. It should not be possible to exhibit a counter-example of a quadratic or cubic system of equations with similar characteristics where our method fails.
5. We will try to be conservative in our estimations and we will try to make our lives harder in order to obtain solid and robust attacks.
   The approach of this paper is conservative and could possibly later be discovered to be an overkill, with respect to events such as additional degree falls to make such attack work easier than expected. We simply do not want to rely on events we don't understand.

3.2 On F/T Ratio, Equations Topology Density and Connectivity Criteria

In this paper we propose another major general paradigm or philosophy which allows to achieve the objectives above in a quite specific way.

1. We want to design systems of equations which are able to achieve certain characteristics in terms of density, topology, and connectivity where the equations are viewed in terms of graphs or hypegraphs in which various types of equations connect different types of variables.
   We want to have high connectivity not only in general or in average case, but for all non-linear monomials we use.
2. We want to obtain equations which are very largely overdefined [6].
   More precisely we would like to have a very large R/T ratio [18, 21, 19] or the F/T ratio, where F is the the number of linearly independent equations.

We will frequently postulate that systems with the characteristics above plus some additional conditions are likely to become efficiently solvable. However frequently it will not yet sufficient to obtain equations which in fact ARE solvable. This is major difficulty in ECDL research which no one was yet bee able to solve in a satisfactory way. We will go back to this question in Section 6.7 and few more times elsewhere.

[6] It could be for example a T' method cf. [19] slides 95-101 which was introduced at Asiacrypt 2002 [21] which however works well primarily for systems of equations over $\mathbb{F}_2$.

3.3 The Tale of Two Parameters

In all our new methods described in this paper there are two parameters, n which is the size of the initial problem, and a parameter K which we will chose to be sufficiently large so that our system of equations becomes efficiently solvable at degree 2 or 3 and by a certain specific method.

Moreover if for a certain value of parameter K the attack does not work, the attacker can increase the value of this parameter. This parameter K should also offer some extra flexibility which allows to optimize the attacks and test different software solvers.

Subsequently we are going to analyse how large K needs to be when n grows and argue that when n is fixed and K increases the number of newly generated independent equations will grow asymptotically quite fast.

This will be **a lot faster** than the number of [new+old] variables, which will make our equations quite massively overdefined. In fact the number of equations R will grow either roughly at the same speed and in some cases even faster(!), than the number of newly introduced monomials [7] T. Thus we will be able to produce a very highly overdefined system of equations with high R/T ratio but also with high density/connectivity characteristics. We will argue that modulo some important and strictly necessary requirement to have a limited number of solutions, such systems are bound to easily solvable in polynomial time. Similarly we will then consider that in order to achieve a "saturation" or "phase transition" point it should be probably sufficient for K to grow polynomially in n.

[7] In this paper, by convention, we do not count the constant monomial in T.

3.4 On Two Major Philosophies In Solving Non-Linear Equations

There are two major philosophies in algebraic cryptanalysis and for the general problem of solving large system of non-linear polynomial/algebraic equations.

1. Either we expand the number of variables.
2. Or we expand the number of monomials.

Both types of methods already existed and both philosophies worked quite well in their own (somewhat disjoint) space in algebraic cryptanalysis of DES [5]. Both have also been studied for solving systems of polynomial equations over finite finite fields at Eurocrypt 2000 [6]. we have re-linearization technique vs. XL algorithm, cf. [6]. At Eurocrypt 2000 it was concluded that re-linearization technique is highly redundant and that XL works better [6]. Then we discover that at higher degrees XL is also redundant[8].

Let us also recall what is the main working principle in both types of techniques: we make two values grow, yet one grows faster. We will see a lot of examples of this in this paper. It allows one to understand why both families of techniques may and will work.

1. When we add **new monomials**, we grow both the number of monomials T and the number of new equations R. For any system of equations with a certain R/T we can easily improve the R/T ratio by increasing the degree. Then R grows **faster** because there are several ways to obtain the same monomial, cf. slide 80 of [19].
2. When we add **new variables** we also grow both the number of monomials T and the number of new equations T. Here is also R can grow **faster** and sometimes even asymptotically faster than T, which is demonstrated in the present paper, cf. for example Thm. 12.3.1 page 41.

It is important to see that techniques of type 1. expand monomials are nowadays standard, well studied, fully automated by software and do not require a lot of attention. The second family has not been sufficiently studied. It gives the code breaker a **very considerable degree of freedom** which is actually a big a problem: it is not clear **how to even start** to design an attack based on this idea. One major family of techniques are the **EC Codes**, cf. Section 3.7 below.

It is important to note that both approaches 1. and 2. can and should be combined. To put it simply, the second approach makes the first approach work better, equations become more overdefined and the so called degree of regularity is expected to decrease, cf. Section 1. In this paper we are going to propose several new techniques in this space and we postulate that just by applying the EC code technique **the regularity degree can be reduced[9] down to a**

[8] It is in general difficult to remove ALL redundancies in linear dependencies in expanded equations. cf. [3, 21, 22, 8, 9, 7, 1]. Gröbner basis techniques are precisely about removing even more redundancies in XL, cf. [1, 2]. However redundancies are NOT necessarily a problem, simpler or alternative approaches can in FACT be equally fast AND use less memory than some advanced Gröbner basis techniques such as F5.

[9] This however has a price, a lot more new variables and new equations are generated.

really low value such as 2 or 3. The crucial question here is the study of methods to achieve this. We would like to achieve some sort of combinatorial explosion in the equations or/and degree falls generated, similar as in ElimLin algorithm. One methods to achieve starts with a construction where $R/T \to \infty$, similar to super-linear growth for linear variables observed in ElimLin algorithm, not a fiction, cf. later Fig. 1. This will in many interesting cases lead to $F/T \to 1$, where $F \leq R$ is the number of equations which are actually linearly independent and not redundant. In this paper we present several new non-trivial techniques to achieve this objective.

3.5 On Equivalence of Two Approaches/Philosophies

One of the key points in the aforementioned Eurocrypt 2000 paper is that the approach with adding extra variables can be reduced to the XL-like approach, cf. [6]. We expect that it is the same here and that we have either equivalence of the two approaches or a semi-efficient reduction in at least one direction. Here are two examples of general results which show that the two approaches are closely related.

Example 1 - Univariate Polynomials

Theorem 3.5.1. Informally, for one of the natural linear EC code with expansion factor being at least K and which we define later and use extensively in this paper, and **for every degree N** $\in \mathbb{N}$, there exists an equation of type: for every $K' > K_0 \in \mathcal{O}(N)$

$$x1^N \text{AffineSum}(\text{up to } K' \text{ variables excluding } x1) =$$

$$\sum (\text{ monomials of degree up to 2 excluding } x1)$$

The particular form of this equations becomes easier to understand if we remark that this equation becomes quadratic if we consider that $x1^N$ is a new variable. We refer to Thm. C.0.2 for a more precise formulation and in Section A.3 we give an example of how such equations can be generated by explicit closed formulas for specific EC Codes we use.

Example 2 - Multivariate Polynomials

Another nice result in this direction is outlined in Section B for arbitrarily high degrees and studied in detail for cubic monomials in Section 11.1. The main result is quite strong: every monomial of any degree can be eliminated by a certain type of equation. More precisely for EVERY number of distinct variables in our linear EC Code there exists a unique equation such that it contains this product and all the other monomials are of lower degree and their number is linear in the degree [not exponential as the reader might expect], cf. Section 11.1. In particular we have MQ3 equations and Thm. 13.9.4 for eliminating arbitrary cubic monomials and Thm. 11.1.1 contains explicit formulas which allow one to do so.

3.6 Limits of The Equivalence

It is possible that the equivalence will work only in one direction, and the other direction will not be as efficient. Moreover even one direction is not going to be efficient in our opinion, or would require one to develop a lot of extra highly specialized code to integrate in existing Gröbner basis software.

The key point is that our work contains many very specialized equations and though these equations are in theory equivalent ot some degree falls or mutants which could appear in a Gröbner basis computation, we can generate them DIRECTLY, without lengthy computations with long polynomial equations, due to existence of specialized formulas [dedicated algebraic shortcuts]. Moreover these equations can greatly simplify arithmetic in the ring multivariate polynomials modulo the ideal generated by our equations. Essentially some monomials can be replaced directly by relatively simple polynomials. An excellent example which illustrates this principle is again the Example 2 above, each monomial with 3 variables can be replaced by a simple quadratic polynomial which is computed using a direct explicit formula given in Section 11.1.

Remark. These dedicated algebraic shortcuts are not equivalent to precomputations in algebraic cryptanalysis, because these formulas are generic exist in vast numbers of instantiations and accordingly do NOT require storage or access to storage in order to be used for eliminating various degree 3 monomials on the fly when required.

3.7 On EC Codes

The philosophy of adding new variables and in particular THE particular way in which WE understand and implement this philosophy in this paper, can be described as and can be explained in terms of **EC codes**.

Definition 3.7.1 (EC Code).
We call an EC code any injective application

$$F : E(GF(P))^K \to E(GF(P))^K$$

which is defined for all except a small number of special EC points.

Many problems which we study in this paper and related literature are closely related and can be seen as variants of traditional **decoding problems** for special types of EC Codes which are **linear over the EC** but not linear over $GF(P)$.

For example if we want to split a point in 2 $Q = P1 + P2$ with $P1, P2 \in$ a certain subset of $GF(P)$, we can define an EC Code as follows:

$$(x, y) \mapsto [\ (x, y),\ Q - (x, y)\ ]$$

And the goal of the attacker will be to find a small codeword of type

$$[\ (x, .),\ (x', .))\ ]$$

such that x and x' belong simultaneously to a certain subspace, for example such that, as proposed in early work of Semaev in [39], we have simultaneously:

$$x < P^{\frac{1}{2}} \quad AND \quad x' < P^{\frac{1}{2}}$$

Remark 1. In both cases, in error correcting codes and here, the goal can be potentially the same: find a code with a fast/efficient deciding algorithm which also will be a fast point splitting algorithm.

Remark 2. A key advantage here is that the attacker can freely chose the metric by which this decoding can be done, and that solving this decoding problem for more or less any[10] metric or subset, will lead to an index calculus algorithm for the ECDL problem. We refer to [35] for explanations regarding how the point splitting problem relates to the ECDL problem and we refer to [16] to see that NOT all point splitting problems lead to efficient algorithms for solving the ECDL problem.

[10] For as long as it does not depend too much on particular points we want to split.

4 Example to Imitate - ElimLin Connection

There is precedent for the sort of attack we are looking for. It is the behavior of ElimLin algorithm in block cipher cryptanalysis [5, 19, 8, 9, 11]. It is an incredible landmark result in cryptanalysis showing how the "regularity degree" is not at all a limitation in algebraic cryptanalysis, or how it can be defeated by the attacker, and it can be reduced to the lowest possible value with very little effort.

ElimLin is a curious sort of attack, cf. slide 126 in [19]. It can be described informally in 2 simple steps:

1. Find linear equations in the linear span.
2. Eliminate some variables, and iterate (try 1. again).

ElimLin is a stand-alone attack which allows one to recover the secret key of many block ciphers [8, 9, 12, 20] and more recently in [23, 37, 24].

The main characteristic of ElimLin is that it quietly dissolves non-linear equations and generates linear equations. This algorithm basically makes progressively disappear the main and **the** only thing which makes cryptographic schemes not broken by simple linear algebra: non-linearity. It is not clear however why this works and how well the ElimLin attack scales for larger systems of equations. For example in recent 2015 work of Raddum we discover that (experimentally) ElimLin breaks up to 16 rounds of Simon cipher [37] however it is hard to know exactly what happens for 17 rounds.

4.1 A Surprising Phenomenon within ElimLin

Here is a simple observation which is is crucial for understanding why ElimLin algorithms does eventually work and is able to cryptanalyse many block ciphers.

Conjecture 4.1.1. Consider a system of multivariate equations derived from a block cipher written following one of the two basic strategies described in [5, 19]. Consider a simple known plaintext attack with K Plaintext/ciphertext (P/C) pairs. Consider a case such that the cipher is broken by ElimLin, cf. [8, 9, 12, 20, 37]. The number of new and linearly independent linear equations generated by ElimLin algorithm grows faster than linearly with K until it reaches a saturation stage where the cipher is broken by ElimLin.

This claim may seem to be too good to be true. Therefore we are going to illustrate it with a real-life examples extracted from the work of our students and published in [11].

4.2 On Asymptotic Behavior of ElimLin when K Grows

One (old) example which shows that the number of equations grows faster than linear as a function of the data complexity K in ElimLin can be found at slide 153 in [19] which example is from 2006-7 and originally comes from [12].

More examples can be easily obtained using a basic software setup which we use at UCL to run a hands-on student lab session on algebraic cryptanalysis of block ciphers [20], which is part of GA18 course on cryptanalysis taught at UCL. One example could be easily obtained for the CTC2 cipher, cf. [8, 9, 20]. A more

"modern" example can be studied with the recent NSA block cipher Simon. We have used the equations generator for the Simon block cipher developed by Guangyan Song and UCL InfoSec M.Sc. student Ilyas Azeem, the complete source code of which is available at github, cf. [20, 10].

The ElimLin is executed using using one of our implementations of ElimLin [20, 10] which has the nice particularity to display on screen the number of linear equations generated at each stage/iteration of the algorithm (it is the only implementation we are aware of which displays it).

On the figure below we show the number of linear equations generated at stage 4 [counting from 0] of the ElimLin algorithm for 8 rounds of Simon block cipher. We should note that nothing remarkable happens at earlier stages 0,1,2,3, the growth is linear.

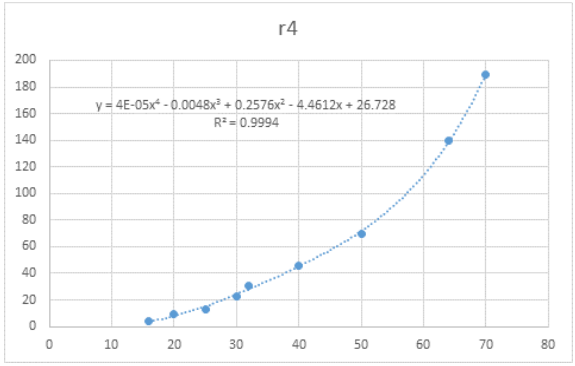

Fig. 1. Number of linearly independent equations generated at stage 4 of the ElimLin algorithm for 8 rounds of Simon 64/128 obtained with the exact software setup of [20].

This graph was generated by Iason Papapanagiotakis-Bousy, UCL Information Security M.Sc. student as part of his individual cryptanalysis project. Our paper on this topic will be published at SECRYPT 2016 conference this summer, cf. [11]. On the picture we also provide the best polynomial approximation at degree 4 which minimizes the squared error and which was computed using Microsoft Excel. It is also noteworthy that with these techniques we have been able to produce extremely accurate predictions which give exact results up to 100 % of the time and which have R^2 close to 1.

4.3 The Unstoppable Force of An Asymptotic

Our conclusion is as follows. We observe that ElimLin breaks the cipher because the number of newly generated linear equations grows **asymptotically faster** than the number of variables. Eventually we obtain a sufficient number of linear

equations which makes ElimLin compute all the variables and obtain the 128-bit secret key together with all the intermediate variables.

What's New, Growth vs. Penetration. This fact was known for at least and decade, though never yet publicly stated in this way as it seems to the author. It was previously observed in different cases where ElimLin was applied cf. slide 153 in [19] and [12] and [8, 9, 5, 20]. In the table at slide 153 in [19] we also observe that as the number of new linear equations grows, these equations penetrate more and more deeply inside the cipher for $2, 3, 5, \ldots$ rounds. We expect the same to happen for other block ciphers when ElimLin was applied. Some sort of converse is also true, it is possible to observe that for all ElimLin runs until a certain threshold K no single new linearly equation which involves the variables in the middle of the cipher is generated. The ElimLin attack really penetrates consecutive round of ciphers one by one from both sides in several clear-cut-threshold steps.

Can This Fail. We do not know any counter-example which would contradict this "super-linear" growth rule. For 8 rounds of Simon cipher we observed that the growth remains strictly linear for equations computed at stage 3, however some equations are only computed at stage 4 of the attack and this occurs quite early starting from $K = 1$. However Simon is a particularly simple cipher. For more complex ciphers, the ElimLin algorithm could have serious problems to enter this behavior or start working and exhibit any sort of non-trivial behavior. However when ElimLin starts to work for some cipher and produces new equations which depend on the plaintext or ciphertext data in a non-trivial way, and already have this "super-linear" character (quite early) it seems that nothing can stop it, we just need to increase K.
This except maybe if there isn't enough data available. A certain type of counter-example could occur for some ciphers with small blocks which would maybe not be able to allow K to be large enough for ElimLin to terminate.

Improvements. We stress that this powerful phenomenon of **combinatorial explosion** which eventually leads to ElimLin breaking the cipher occurs already for a basic straightforward application of ElimLin in a known-plaintext attack (KPA) scenario. Several methods to make ElimLin work better by using well-chosen P/C pairs. One method which has been used ever since ElimLin was invented [8, 9, 5] is to exploit a CPA (chosen-plaintext attacks) with plaintext which differ by extremely few bits, for example in a counter mode. More recently new methods have been proposed in the literature which are able to generate additional linear equations not automatically discovered by ElimLin [23] and [24]. Numerous researchers have demonstrated in their experiments that ElimLin combines very well with equations on bits which can be obtain from any of linear, differential, truncated-differential and cube-like attacks. It is the therefore a highly versatile approach susceptible of many advanced variants and optimizations.

4.4 Lessons Learned

Similarly as with ElimLin, we would like to design an attack on the ECDL problem with a parameter K, such that we can make our system of equations progressively "easier" to solve when we increase K.

Phase transitions. It is known that many NP-hard problems are subject to "phase transition", with certain parameters that problem is hard, and then will rather abruptly transition from "hard" to "easy to solve". This what we observe with ElimLin here. We would like to be able to engineer such a "phase transition" for the ECDL problem in elliptic curves. This is not going to be an easy task.

5 Summation Polynomials in Elliptic Curves

A majority of works on this topic consider primarily and exclusively binary elliptic curves. We know only two exceptions to this rule: the old initial Semaev paper [39] and one of the most recent papers [35]. In this paper we consider exclusively ordinary elliptic curves modulo P such as used in real-life applications on the Internet, in payment systems such as bitcoin, etc.

5.1 Elliptic Curves Mod P

One of the most popular ways to define an elliptic curve group is to use the Weierstrass equation:

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

In addition in large characteristic > 3 it is typically assumed (without any loss of generality, cf. Section 3.1.1. in [28]) that we have $a_1 = a_3 = a_2 = 0$ and $A = a_4$ and $B = a_6$ so we get:

$$Y^2 = X^3 + AX + B$$

In contrast for typical curves in characteristic 2 we assume (following [38]) that $a_1 = 1$, $a_3 = 0$, $a_4 = 0$, and and $A = a_4$ and $B = a_6$ so we get:

$$Y^2 + XY = X^3 + AX^2 + B$$

5.2 Summation Polynomials

Now we consider a summation of 3 points on the elliptic curve:

$$(x_1, y_1) + (x_2, y_2) + (x_3, y_3) = \infty$$

We consider the case of characteristic > 3. Following [32] we have:

$$S_3(x_1, x_2, x_3) =$$
$$(x_1^2x_3^2 + x_2^2x_3^2 + x_1^2x_2^2) - 2(x_1^2x_2x_3 + x_1x_2^2x_3 + x_1x_2x_3^2) - d_2(x_1x_2x_3) +$$
$$-d_4(x_1x_3 + x_2x_3 + x_1x_2) - d_6(x_1 + x_2 + x_3) - d_8$$

where $d_2 = a_1^2 + 4a_2$ $d_4 = 2a_4 + a_1a_3$ $d_6 = a_3^2 + 4a_6$ $d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$ also following [32, 25].

Therefore when we assume that $a_1 = a_3 = a_2 = 0$ and $A = a_4$ and $B = a_6$ we get:

$$S_3(x_1, x_2, x_3) =$$
$$(x_1^2x_3^2 + x_2^2x_3^2 + x_1^2x_2^2) - 2(x_1^2x_2x_3 + x_1x_2^2x_3 + x_1x_2x_3^2) +$$
$$-2A(x_1x_3 + x_2x_3 + x_1x_2) + A^2 - 4B(x_1 + x_2 + x_3)$$

Which following [38] can also be written as:

$$S_3(x_1, x_2, x_3) = (x_1 - x_2)^2x_3^2 - 2\left[(x_1 + x_2)(x_1x_2 + A) + 2B\right]x_3 + (x_1x_2 - A)^2 - 4B(x_1 + x_2)$$

5.3 Point Splitting Polynomials

In order to design an index calculus algorithm for the ECDL problem, we need to solve the problem of point splitting. Research research shows that splitting in 2 or splitting in 3 is easier than expected [16], at least for binary elliptic curves. It is easy to see that in order to obtain any improvement on Pollard's Rho complexity of $2^{n/2}$ we need to be able to split a point in at least 5.

Let $R = (R_X, R_Y)$ be the target point on the elliptic curve which we want to split in a form

$$R = P_1 + \ldots P_i + \ldots + P_M$$

with some M points P_i $i = 1..M$.

5.4 Point Splitting In Two Using S3

The most basic problem which we would like to be able to solve using the Semaev polynomials is the problem of splitting a point in two:

$$S_3(P_{1X}, P_{2X}, R_X)$$

where P_1 and P_2 should lie in a well-chosen subspace of size $2^{n/2}$.

5.5 General Point Splitting

We consider the problem of splitting in M points using the polynomial S_{M+1}:

$$S_{M+1}(P_{1X}, P_{2X}, \ldots, P_{MX}, R_X)$$

where the P_i should lie in a well-chosen subspace of size $2^{n/M}$.
This is the initial Semaev approach from 2004 in [39].

5.6 Degree Considerations

It is possible to see that the degree of this polynomial S_{M+1} is 2^{M-1} in each variable [39] and the total degree is $(M+1)2^{M-1}$. We formalize this as follows:

Definition 5.6.1 (r-degree).
We say that a polynomial belongs to the set of polynomials of r-degree s if it is a sum of products of up to s powers $1..r$ of the individual variables.

According to this definition the Semaev polynomial S_{M+1} has 2^{M-1}-degree equal to $M + 1$.

5.7 Point Splitting In M Parts Using S3 Polynomials

More recently in 2015 a particular way to re-write this problem using only the simplest non-trivial summation polynomial S_3 and with many additional variables have been proposed by Semaev in 2015, cf. [38]. We recall this particular way to encode the problem of splitting the point on elliptic curve. We call x_i the x coordinate of point P_i in $GF(2^n)$ and let u_i be $M-2$ auxiliary variables in $GF(2^n)$.

$$\begin{cases} S_3(u_1, x_1, x_2) \\ S_3(u_1, u_2, x_3) \\ S_3(u_2, u_3, x_4) \\ \vdots \\ S_3(u_i, u_{i+1}, x_{i+2}) \\ \vdots \\ S_3(u_{M-3}, u_{M-2}, x_{M-1}) \\ S_3(u_{M-2}, x_M, R_X). \end{cases}$$

We have $M-1$ equations in $GF(2^n)$ where M is the number of points in our decomposition of R as a sum M elliptic curve points. We can call it a Semaev-serial system of equations as it effectively is **a serial connection**[11] of several systems of equations of type S_3 in a certain encoding (with a topology of straight line with connections only between consecutive components).

5.8 On Topology of Equations

Block ciphers are typically quite hard to break by algebraic attacks, and this sort of **block cipher or serial topology** of [38] is an example of how **not** to approach this problem. An example of alternative approach can be found in [30] which paper proposes to decompose S_M into several S3 equations using **a tree topology**, see Section 5 of [30]. We believe nevertheless that this approach from 2015 and also the one in [30] is better than the initial one from 2004 in [39], which leads to polynomials of higher degree and overall larger and more complex systems of equations. This point is also made in the abstract of [30]. Another approach which has this unfortunate serial topology is the very recent approach to constraints for certain elliptic curves mod P in [35]. In this paper we will propose an approach which at antipodes w.r.t. a serial connection topology. Our goal is to achieve some sort of **dense topology** where there is a lot interaction between different equations and different monomials. We believe that this is actually the only plausible way to achieve something which is efficiently solvable.

[11] This sort of topology for systems of equations is very common in block cipher cryptanalysis [5, 8, 13] and the hardness of solving these systems can be seen as a hardness to derive ANY sort of algebraic or statistical knowledge about the middle variables not easily accessible to the attacker.

Part II

High-Level General Point Splitting Methods

6 High Density Approach For General Point Splitting

In this section we propose our first method and approach which leads to **highly overdefined** systems of equations. Moreover we will also try to construct equations which have some sort of very dense[12] topology. Our construction is completely geeneral: we work for curves mod P and consider splitting in an arbitrary number of components. Later we will at some special cases such as splitting in two.

6.1 Our Main General Construction Ex_M^K

As in [38] we are going to add additional variables which is a frequently used trick in algebraic cryptanalysis. We are going to present a general approach for any M and ordinary elliptic curves mod P. We will generates a certain highly connected and highly overdefined system of equations which later will be called Ex_M^K.

We want to solve:

$$S_{M+1}(P_{1X}, P_{2X}, \ldots, P_{MX}, R_X) = 0$$

where the P_i should lie in a well-chosen subspace.

6.2 Adding Variables Using A Linear EC Code

We are going to consider K constant points S_i which lie on the same elliptic curve In this paper we are going to assume that these points are random, and that their X coordinates are all distinct numbers mod P. Moreover the first point will be always assumed to be $S_0 = \infty$. In future works we are going to explore how to improve our attacks by using a well-chosen set of points S_i.

We are going to add new variables for the X coordinate of every possible $P_i + S_j$ which addition is done on the curve. Let

$$Z_{ij} = (P_i + S_j)_X$$

We have assumed $S_0 = \infty$ so that the original P_i is also one of the these variables with $\forall i P_i = Z_{i0}$.

Our new equations are going to be:

$$S_{M+1}(Z_{1i_1}, Z_{2i_2}, \ldots, Z_{Mi_M}, \ (R+\sum_{j=1}^{M} S_{i_j})_X) = 0 \quad \forall i_j \in \{0, \ldots, K-1\} \qquad (Ex_M^K)$$

We call these expanded equations Ex_M^K.

[12] Informally, dense topology is the contrary of the serial or block cipher topology in [38, 35] Very dense could be defined as something like: "one equation shares many major non-linear monomials with a number of other equations as large as possible" where in our approach "as large as possible" means a number higher than a constant, for example $\mathcal{O}(K)$.

6.3 Comparison to EC Codes

In our equations we have a large linear EC Code with expansion factor of $K+K$ times defined as follows:

$$P \mapsto [\ P_1 + S_0,\ \ P_1 + S_1,\ \ \ldots\ P_1 + S_{K-1};\quad P_2 + S_0,\ \ P_2 + S_1,\ \ \ldots\ P_2 + S_{K-1}\ ]$$

and our equations are Semaev polynomials after substitution of one variable by specific constants computed on the EC. So they are not exactly just[13] Semaev equations.

6.4 Analysis of Equations $\boldsymbol{Ex_M^K}$

Here in each equation the last number is a constant which the attacker computes. This decreases the degree of the equations and it is THE place where the law of the elliptic curve interacts with the construction of our equations in a very substantial[14] way.

We have

$$R = K^M$$

equations of degree 2^{M-1} in each variable and one variable is a constant. Here according to Def. 5.6.1 each equation has 2^{M-1}-degree equal to M and total degree up to $M2^{M-1}$.

The total number of monomials which appear in these equations is approximately:

$$T = \mathcal{O}(K^M)$$

More precisely if we ignore products of $M-1$ or less powers of variables, and only look at the dominating part with 2^{M-1}-degree equal to exactly M, it is about

$$T \approx 2^{M(M-1)} \cdot K^M / M!$$

This is a very largely overdefined system of equations, as $T \gg V$ where V is the number of variables which grows only linearly with K:

$$V = M \cdot K$$

We recall that M is a constant and K is allowed to grow in an arbitrary way.

[13] If the EC law was a black box group and the results of long point additions were essentially random rather than related to each other, not that we believe any such claim, we could consider that our equations are to a large degree random equations with some specific monomials, or at least locally they should look like random polynomials.

[14] This interaction satisfies our informal "dense topology" requirement in a certain different way than in other places and qualifies to be called a "densely connected" method. Different powers of the constants obtained form a complex EC point addition are used and their effect is diffused very substantially due to the fact that each monomial of 2^{M-1}-degree equal to M can be obtained from up to 2^{M-1} different monomials of 2^{M-1}-degree equal to $M+1$ when one out of M variables is replaced by a constant in a Semaev polynomial S_M.

6.5 On Linear Dependencies

Linear dependencies are a major difficulty in algebraic cryptanalysis and a major topic of study in Gröbner basis theory. They make that many attacks do not work as well as expected, cf. [3, 21, 22, 8, 9, 7, 1] and many other.

At this stage we do NOT have this problem.

Theorem 6.5.1. 100% of equations in set Ex_M^K are **linearly independent** and we have

$$F = R = K^M.$$

Proof: Each of our equations contains a monomial

$$Z_{1i_1}^{2^{M-1}} \cdot Z_{2i_2}^{2^{M-1}} \cdot \ldots \cdot Z_{Mi_M}^{2^{M-1}}$$

which monomial appears exactly once, only in this equation and in no other equation in our set Ex_M^K .

6.6 On R/T and F/T Ratio

In algebraic cryptanalysis of block ciphers R/T ratio is frequently studied. Initially block ciphers cryptanalysts aim at a ratio of type say $R/T = 1/3$, or $1/4$ and as large as possible see slides 64,69-70 in [19] and [21, 22]. Then the expansion step such XL,XSL or other algorithm improves the ratio and makes it closer to 1, cf. slide 73 in [19]. In this paper we simply propose yet another **new** and original method to make this R/T ratio grow and which does not expand the degree of the equations compared to one single polynomial or type S_M. This degree remains constant at all times. Instead we expand the number of variables by adding new variables and new equations.

For now the original R/T ratio is:

$$R/T \approx M!/2^{M(M-1)}$$

And following Thm. 6.5.1:

$$F/T = R/T \approx M!/2^{M(M-1)}$$

This is a constant in any point splitting problem with M parts.

6.7 The Philosophy of Ex_M^K Approach

As in many algebraic attacks approaches we aim at obtaining degree falls and constructing an efficiently solvable overdefined system of algebraic equations. However we expand the equations NOT by increasing the degree but by increasing the number of variables[15] following a linear EC Code expansion.

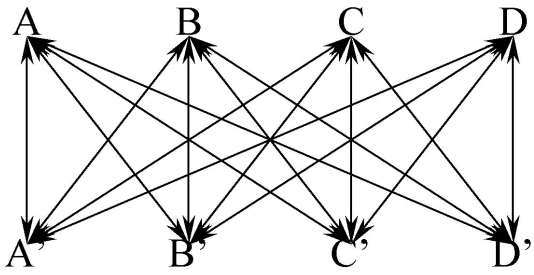

Fig. 2. Graph of connections between points on both sides, $M = 2, K = 4$

This serves our two main strategic goals:

1. We will obtain systems of equations which are **massively overdefined** with a very high R/T ratio which is substantially higher than in previous works on this topic, cf. Section 6.9 below.
2. We conserve a special structure which makes that only some types of monomials are used during the attack, however we try to make sure to obtain a very **densely connected topology**.

In this paper we will argue that the combination of these 2 properties can make systems of equations efficiently solvable even though it is clear that these conditions are NOT sufficient, see later Section 6.9 and 12.8.

6.8 On [Complete] Graphs and HyperGraphs

When $M = 2$ we can study our equations in terms of graphs. If we connect in a graph each pair of variables for which we wrote an S3 equations with a third variable being a constant, we get a bi-clique graph, cf. Fig. 2. Later in this paper we will also add connections on each side, see Section 7 and later cf. Fig. 5 page 42 and we will obtain truly complete graphs which will somewhat lose their bi-partite structure (or it will be less visible).

In general for any $M \geq 2$ our approach can be studied in terms of [complete] hypergraphs in which hyper-edges can connect up to M vertices which correspond to distinct variables in our system of equations.

[15] This is again following one of so called "Fast" algebraic attacks strategies, cf. slide 84 in [17] and slide 121 in [19] and specifically avoiding expansion of the degree of the equations, cf. slides 82-86 of [17]. This can be seen as the dual approach of the degree expansion approach which has dominated this space since 2000 [6].

6.9 On F/T Ratio and Solvability

We claim that a ratio F/T being a constant makes our equations Ex_M^K potentially solvable by Gröbner basis algorithms. A constant ratio F/T should be compared to a much smaller ratio of type $1/n^d$ which is the case in ALL previously published papers on this topic [39, 38, 36, 35] and many other.

In general however it is known that **no amount of work on** R/T or F/T can solve algebraic equations **if they have a large number of solutions**, which is the case for our equations Ex_M^K. For algebraic equations it would be sufficient to guess some variables and decrease the number of solutions, cf. FXL algorithm [6, 7]. In the case of equations mod P this is a major difficulty and no satisfactory[16] solution to this problem is known.

6.10 Sub-Exponential Attacks On EC Discrete Log Problem?

In spite of these difficulties let use assume that there exists[17] some approach which allows to reduce the number of solutions. Then probably with our equations Ex_M^K we can solve our point splitting problem:

Conjecture 6.10.1. Assume that the point splitting problem is encoded by using the same set of monomials as used in our equations Ex_M^K [this assumption is probably NOT realistic[17]]. Then one should be able to prove in Gröbner basis theory and the combined system of equations can be solved in sub-exponential time by F4 or similar Gröbner basis algorithm [1, 2].

Justification: For any system of equations with $R/T = 1/8$ we can easily improve the R/T ratio by increasing the degree. The basic mechanism is widely known: R grows and T grows but R grows faster because there are several ways to obtain the same monomial, cf. slide 80 of [19]. There are also questions of linear dependencies in expanded equations, cf. [3, 21, 22, 8, 9, 7, 1].

Here we need to design and analyse a custom version of this process adapted to the special structure of our monomials. We are going to multiply each monomial of 2^{M-1}-degree of up to M, by all possible powers T^r for $r = 1..2^{M-1}$ and for any variable T which does NOT appear in this monomial. Thus we avoid ever creating powers higher than 2^{M-1} and we move to equations of 2^{M-1}-degree of $M + 1$.

Most of the time a variable T does not appear in our monomial and we have $V = MK$ variables. The number of expanded equations is now about :

$$R^{expanded} \approx 2^{M-1} \cdot M \cdot K^{M+1}$$

and they are still of degree up to 2^{M-1} in each variable. We count the total number of monomials which appear in these equations. All monomials have 2^{M-1}-degree equal to at most $M + 1$, se we have approximately:

$$T^{expanded} \approx 2^{(M+1)(M-1)} \cdot K^{M+1}/(M + 1)! = \mathcal{O}(K^{M+1})$$

Initially we had $R/T \approx M!/2^{M(M-1)}$. Now we have:

[16] Two major very recent attempts to solve this problem can be found in [35] however they do not have a "dense topology" we are trying to achieve in this paper.

[17] The next best thing is probably again the method(s) of [35].

$$\frac{R^{expanded}}{T^{expanded}} \approx \frac{2^{M-1} \cdot M \cdot (M+1)!}{2^{(M+1)(M-1)}} \approx M(M+1) \cdot (R/T)$$

We see that our ratio will improve roughly M^2 times (we neglected terms of lower r-degree). this at the price of increasing the 2^{M-1}-degree by 1. Eventually after a few steps we will achieve $R/T \approx 1$, though by the present method we do not claim that it will become exactly equal to 1.

Polynomial? Subexponential? - future work: We conjecture that a family of systems of equations which achieve $F/T \to 1$ and **have a unique**[18] **solution** is efficiently solvable in the worst case and that the solution can be computed in polynomial time.

This also for a peculiar set of monomials we have here in a similar way as in general, for general dense polynomials of a certain degree [1]. The situation is quite clear for equations over $\mathbb{F}_2$ with the so called T' method which was proposed at Asiacrypt 2002, cf. [19] slides 95-101 and [21].

For larger fields the result is more hazardous. We will leave this question for future research.

Remark: Later in this paper we will show how to achieve $F/T \approx 1$ **directly** without additional degree expansion [needed so far].

[18] Unhappily we don't achieve this here.

7 High Density Approach for $M = 2$

In this section we are going to re-state approach of building a highly overdefined system of equations with dense topology for $M = 2$ in particular. We are also going to add more equations. Previous equations Ex_2^K formed a bi-clique graph, cf. Fig. 2 page 28. Now we will add connections internal to each side and we will obtain a complete graph, cf. Fig. 5 which figure we also reproduce on this page for better readability. The new equations will be called Dx_2^K.

8 Summary of Ex_2^K and New Equations Dx_2^K

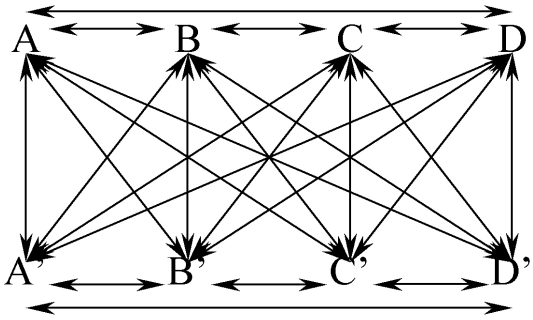

Fig. 3. Graph of connections in Ex_2^K and in new equations Dx_2^K

We are now going to re-state and summarize our how previous equations are created and we will introduce new additional equations. We want to solve:

$$S_3(P_{1X}, P_{2X}, R_X) = 0$$

where the P_i lie in some subspace of size $2^{n/M}$ which we do not yet specify.

Again we consider K constant [random] EC points S_i with distinct X coordinates and $S_0 = \infty$ and let:

$$Z_{ij} = (P_i + S_j)_X$$

We have $V = K + K$ variables. We consider two sets of equations where the first is the same as before for $M = 2$.

$$S_3(Z_{1i}, Z_{2j},\ (R + S_i + S_j)_X) = 0 \quad \forall i, j \in \{0, \ldots, K-1\} \qquad (Ex_2^K)$$

As before we have:

$$R^{Ex} = K^2.$$

Here are the additional equations. For any pair of variables Z_{ki} and Z_{kj} we code the fact that the difference on the EC of suitable points is a known constant:

$$S_3(Z_{ki}, Z_{kj}, (S_i - S_j)_X) = 0 \quad \forall k \in \{1, 2\} \forall i, j \in \{0, \ldots, K-1\} \qquad (Dx_2^K)$$

We have new equations of 2-degree 2. Initially we have

$$R^{Dx} = M\binom{K}{2} \approx K^2$$

such equations. New we count the monomials in these equations. First we count monomials of 2-degree 2 which come from the first set Ex_M^K. We have monomials of type TU, T^2U, TU^2, T^2U^2:

$$T^{D2x2} \approx 4\binom{K}{2} \approx 2K^2$$

Now we have a different disjoint set of 2-degree 2 monomials TT' or T^2T' etc from the second set Dx_M^K.

$$T^{E2x2} = 4 \cdot 2\binom{K}{2} \approx 4K^2$$

The number of affine monomials is $V + 1 = 2K + 1$. Overall the total number of monomials which appear in these equations is approximately:

$$T \approx 6K^2$$

Again we expect that:

Theorem 8.0.2. 100 % of equations in set Ex_M^K are linearly independent and we have

$$F = R = 2K^2.$$

Proof: Each of our Ex/Dx equations contains one monomial $Z_{1i}^2 \cdot Z_{2j}^2$ or $Z_{1i}^2 \cdot Z_{1j}^2$ or $Z_{2i}^2 \cdot Z_{2j}^2$ which monomial appears exactly once, and appears in no other equation in our set $Dx_2^K \cup Ex_2^K$.

In this system of equation we initially have

$$R/T \approx \frac{2}{6}$$

As in Section 6.10 we conjecture that this probably leads to a subexponential algorithm IF constraints can be coded without creating extra monomials [which again is rather unrealistic and we will study this question later].

9 On The Role of Semaev Polynomials

Until now the Semaev polynomials play a central role in our efforts to cryptanalyse the ECDL problem. We have just proposed a method to solve one single equation which is a Semaev polynomial by generating a vast number of other Semaev polynomials which leads to a highly overdefined system. In what follows wee are going to discover that there exist other **new polynomial equations** which are simpler the Semaev polynomials, have **lower degree** and lower complexity, and [as it will turn out] are NOT contained in the ideal a Gröbner basis generated by our Semaev polynomials used so far. In addition they can be generated in larger numbers with asymptotically interesting properties, so that for example their number grows asymptotically quite fast and some degree of saturation [as in ElimLin] becomes possible.

Until now with Semaev polynomials cf. Section 6.6 we could achieve by a variable change [replacing each power of a variable by a new variable] a system of pure quadratic equations [still based on Semaev polynomials] with $F/T = \mathcal{O}(1)$ which in practice is a fairly small constant. Our objective is a lot more ambitious, we want to construct a system of purely quadratic equations such that $F/T \to 1$.

Part III

Towards Better / Faster Point Splitting Methods - A Proof Of Concept For $M = 2$

10 Enhanced High Density Approach For $M = 2$

In this section we are going to prove some quite surprising results about our approach. Our current approach with $Dx_2^K \cup Ex_2^K$ can be summarized as follows: we have K variables of type 1 and K variables of type $P2$. Then for any pair either of type 11, 12 or 22 we are able to connect it by an S3 equation of 2-degree 2. We get a sort of complete graph: all connections are present. We are interested in triangles (cycles of length 3) in this graph. Without loss of generality each of them is of the form:

$$(P_1 + S_i)_X,\ (P_1 + S_j)_X,\ (P_2 + S_k)_X,$$

This triangle is in our system of equations coded as 3 equations:

$$\begin{cases} D(ij) & S_3(Z_{1i}, Z_{1j}, (S_i - S_j)_X) = 0 \\ E(ik) & S_3(Z_{1i}, Z_{2k},\ (R + S_i + S_k)_X) = 0 \\ E(jk) & S_3(Z_{1j}, Z_{2k},\ (R + S_j + S_k)_X) = 0 \end{cases}$$

These 3 equations interact in an interesting way. They are characterized by the fact by for any pair either their sum or a difference is known.

Now we have a surprising result:

Theorem 10.0.3. The attacker can essentially in constant time compute 1 additional linearly independent equation which use exactly the same 12 of 2-degree 2 monomials which appear in $D(ij), E(ik), E(jk)$ and no new monomial is needed. which is of total degree 2, i.e. no variable is squared. In fact it has only 3 non-linear monomials $Z_{1i}Z_{1j}, Z_{1i}Z_{2k}, Z_{1j}Z_{1k}$. It has 7 coefficients total which are polynomials in the inputs of the problem which are the fixe sums or differences between the 3 points, one for each side of the triangle in question.

Justification: This fact is a bit surprising because this equation is NOT[19] in the ideal generated by the 3 S3 equations which form the triangle. We will not obtain it from as degree falls in a Gröbner basis computation unless we add additional equations to it.

We thank our students Wei Shao and Huanyu Ma [UCL M.Sc. Information Security 2015-16 and Cryptanalysis GA18] for computing these equations by two different methods.

Definition 10.0.4 (MQ2(x,y,z)).
In what follows are going to call $MQ2(Z_{1i}, Z_{1j}, Z_{2k})$ this particular equation which is of total degree 2.

In Section A.1 we show a realistic example of such an equation and in Section 11 we provide the general result and provided closed formulas for computing this MQ2 equation.

[19] We have verified this independently with three different software tools: SAGE, Maple and proprietary software.

11 MQ2 - A New Method of Generating Quadratic Equations

Now we are going to explain in which exact cases the equations of type MQ2 exist and can be generated.

Let $P \neq \infty$ be a point on an elliptic curve modulo a large prime P and let S_0 and $S_1 \neq S_0$, and $S_2 \neq S_0$ be three distinct constants, i.e. $S_1 \neq S_2$.

We consider a linear EC Code defined as follows:

$$P \mapsto [\ P + S_0,\quad P + S_1,\quad P + S_2]$$

where additions are done on the curve. We call $ix1, ix2, ix3$ the x coordinates of these three points and we expand our constant to a [redundant] set of differences on the EC for which we also consider only x coordinates and ignore the y coordinates, which is another form of EC code which we use here because it symmetric greatly simplifies our equations and our result. By definition we have:

$$\begin{aligned}(ix1, .) &= P + S_0\\ (ix2, .) &= P + S_1\\ (ix3, .) &= P + S_2\\ (dx12, .) &= S_1 - S_0\\ (dx13, .) &= S_2 - S_0\\ (dx23, .) &= S_2 - S_1\end{aligned}$$

Fig. 4. Notation for MQ2 and MQ3 Equations

We call each such configuration of three points with known differences dxij a **triangle** which involves three main variables $(P + S_i)_x$ which are the x coordinates for our EC codewords as defined above where P is a free variable which à priori not known.

Theorem 11.0.5. For every triangle with the assumptions above, there exist a polynomial in 6 variables ixj and $dxij$ and with only 3 non-linear [quadratic] monomials in the ixj which is true each time except when one point is degenerate, i.e. true always if all the three variables satisfy $P + S_i \neq \infty$.

For the bitcoin elliptic curve and any elliptic curve mod P where $A = 0$ and $B = 7$ this equation is exactly as follows.

```
1*(84*dx12*dx13+84*dx12*dx23+84*dx13*dx23)  +
ix1*(56*dx12+56*dx13+84*dx23+dx12^2*dx13^2-dx12^2*dx23^2-dx13^2*dx23^2+2*dx12*dx13*dx23^2)  +
ix2*(56*dx12+84*dx13+56*dx23-dx12^2*dx13^2+dx12^2*dx23^2-dx13^2*dx23^2+2*dx12*dx13^2*dx23)  +
ix3*(56*dx12+28*dx13+28*dx23+2*dx13^2*dx23^2-2*dx12*dx13*dx23^2-2*dx12*dx13^2*dx23)  +
ix1*ix2*(28-2*dx12*dx13^2-2*dx12*dx23^2+2*dx13*dx23^2+2*dx13^2*dx23+4*dx12*dx13*dx23)  +
ix1*ix3*(28+2*dx12*dx23^2-2*dx13*dx23^2-2*dx12^2*dx13+2*dx12^2*dx23+4*dx12*dx13*dx23)  +
ix2*ix3*(28+2*dx12*dx13^2+2*dx12^2*dx13-2*dx12^2*dx23-2*dx13^2*dx23+4*dx12*dx13*dx23) = 0
```

Furthermore, we have a completeness property: this equation is unique modulo the dependencies between that $dxij$ and it is entirely unique after substitution by fixed constants of $dxij$. Across all possible code words in the EC code defined above there is **exactly** one linearly independent equation involving 7 monomials $1, ixi, ixi * ixj$.

In general there exists one single unique equation of this form and for any other elliptic curve mod P in Weierstrass form, we just have a few more terms which depend on A, here $A = 0$ and the equation is simplified.

We also have a similar equation with an 8-th monomial in the xi, which is the product of all the 3 variables. This second equation is called MQ3 and the corresponding formulas are given in below, cf. Thm. 11.1.1.

In addition in Section B page 65 we generalize this equation to generate very similar unique equations of arbitrary degree d. Similar results also exist if we square one variable, cf. for example C.0.2 which equations are also unique equations with certain fixed set of monomials, and we refer to Section A.3 and to Section A.4 for specific examples of how such more general equations can be generated by some other explicit closed formulas for another specific EC Code which is essentially the same except that we allow more monomials which can be seen as extending the EC code by another polynomial-type code built on the top of it.

11.1 MQ3 - A Major Variant of MQ2

We also have:

Theorem 11.1.1. For every triangle with the assumptions above, there exist a second polynomial in 6 variables ixj and $dxij$ and with only 4 non-linear [3 quadratic and cubic] monomials in the ixj are used, again true all the time except when one point is degenerate, i.e. $\forall_i (P + S_i)_x \neq 0$. Again when $A = 0$ this equation is exactly:

```
-4B*(dx12*dx13+dx12*dx23+dx13*dx23)  +
-4B*(ix1+ix2+ix3)*(dx12+dx13+dx23)  +
ix1*ix2*(dx12*dx13^2+dx12*dx23^2-dx13*dx23^2-dx13^2*dx23)  +
ix1*ix3*(-dx12*dx23^2+dx13*dx23^2+dx12^2*dx13-dx12^2*dx23)  +
ix2*ix3*(-dx12*dx13^2-dx12^2*dx13+dx12^2*dx23+dx13^2*dx23)  +
ix1*ix2*ix3*(-dx12*dx13-dx12*dx23-dx13*dx23) = 0
```

Furthermore, we have this equation is also unique after substitution of constants of $dxij$, and across all possible code words in the EC code defined above there is **exactly one** linearly independent equation involving our 8 monomials.

This equation can be used to compute Eliminators: equations which allow to replace terms of degree 3 with lower degree terms, E is the number of Eliminators, cf. Section 13.9.

11.2 On Linear Dependencies, MQ2 vs. MQ3

It is easy to see that, unlike MQ2 equations most of which are typically linearly dependent in applications which use them, which we will see later in Thm. 12.3.1 and many subsequent results, the MQ3 equations are strictly linearly independent. We have the following result:

Theorem 11.2.1. Consider an arbitrary set of K points P_i on an elliptic curve mod P such that for every pair of points we know either their sum or their difference for the EC addition law (by convention we call such set an **EC code**[20] **expansion** of one point). Consider the x coordinates of the points and ignore the y coordinates. We call $D3^K$ the set of equations obtained from writing the equation MQ3 given by the formulas of Section 11.1 above for each triple of points.

We have exactly $\binom{K}{3}$ equations $D3^K$ and 100 % of equations in set $D3^K$ are linearly independent.

$$F = R = 2K^2.$$

Proof: From definition in Section 11.1 it is obvious that each cubic monomial in our $D3^K$ equations appears in exactly one equation, the equation written for that exact triple of variables.

12 Consequences of Using MQ2/MQ3 Equations

It is possible to see that with MQ2 equations we are going to obtain things which were not quite possible to have with standard Semaev polynomials we have used so far.

12.1 What's Special About the New Equations

We claim that our new equations MQ2 are **a lot more interesting** than any other equations we have seen so far, and anything which is contained in the ideal generated by the S3 polynomials. The key remark is that this new equation MQ2 is such that it depends simultaneously on all 3 sides of the triangle. This has very important consequences, basically the number of such equations which can be generated is overall going to be very substantially larger than just multiplying the number of Semaev equations by a constant.

In order to see that, we consider again our previous setup from Section 7 where we now added a lot new and simpler equations cf. Section 10.

12.2 How To Generate a Lot of Equations

The main difference between our MQ2 equation and the 3 initial S3 equations from which we started is that so far the number of (old) equations would grow quadratically with K. The initial 2-degree quadratic equations S3 exist for every pair i, j in the full system of equations $Dx_2^K \cup Ex_2^K$ with $K + K$ variables. At the same time the number of variables also grows quadratically.

Until now the only thing we could hope for and which we have achieved was something like $R/T \approx \frac{2}{6}$. This is already better than many previous approaches in the literature. However this not very satisfactory.

Unhappily current Gröbner basis theory does not provide any real guarantee that such systems of equations could be efficiently solvable. This remains a bold

[20] As always in error-correcting codes, it is a linear code of a certain type where the linearity holds over the elliptic curve.

conjecture and the answer could be that it depends on the constant or the complexity could be a very fast growing sub-exponential implied by the progressive increase in R/T which cf. Section 6.10, which algorithm would be far from being practical to execute.

Things look **a lot better** with our set of pure unique quadratic equations $MQ2(Z_{1i}, Z_{1j}, Z_{2k})$.

12.3 On Combinatorial Explosion Due to MQ2 Equations

We continue the study the new MQ2 equations which can be written for the initial system of equations $Dx_2^K \cup Ex_2^K$ with $K + K$ variables.

Theorem 12.3.1. Our pure quadratic equations $MQ2(t, u, v)$ exist and are obtained in the same way which we detail in Section 11 **for any**[21] **triple of variables** t, u, v out of $K + K$ regardless on which side[21] they are. We call DEx_2^K the set of these equations for all possible $K+K$ variables $t, u, v \in \{Z_{1i}, \ldots, Z_{2i}\}$.

The number of MQ2 equations in DEx_2^K grows as $\mathcal{O}(K^3)$ which is **a lot faster** than the number of monomials which remains $\mathcal{O}(K^2)$.

Proof:

We have:

$$R^{DEx} = \binom{2K}{3} \approx \frac{4}{3}K^3$$

The set of monomials is smaller than ever before, as we do not have any squares anymore for non-linear terms:

$$T^{DEx} \approx \binom{2K}{2} + 2K \approx 2K^2$$

Here we can easily achieve $R/T > 1$. We have **a super highly overdefined** system of equations.

[21] We can make them for any triangle on our graph in which each pair of variables out of $2K$ are connected by S3 polynomials because either their sum or their difference is known, cf. Fig. 5. In each case the quadratic equation MQ2 can be computed by the same formula cf. Section 11. This formula takes as input the sums or differences between each pair and requires only x coordinates and is indifferent whether these are sums or differences. It is also invariant if we replace the 3 points by their opposites on the curve. So that we can apply the same formula and get for example equations of type $MQ2(Z_{2i}, Z_{2j}, Z_{2k})$.

12.4 On Dense Topology and High Graph Connectivity in DEx_2^K Equations

There is no doubt either that the topology of this system of equations is **excessively dense**: each equation shares each non-linear monomial it has with $\mathcal{O}(K)$ other equations which connections go into and spread over about half of the whole set of equations. Our set of variables forms a complete graph with all pairs are connected and which contains a very large number of graph cliques of different sizes (complete subgraphs).

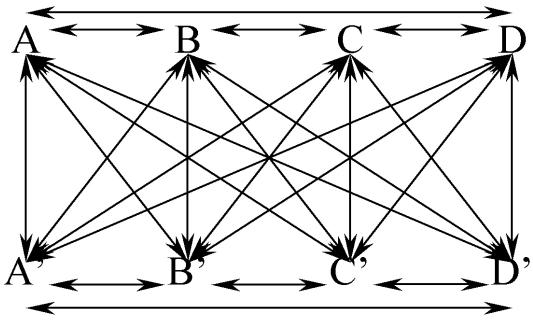

Fig. 5. Graph of connections between points on both sides, $K = 4$

Definition 12.4.1 (Triangles).
We recall that in this paper we will call "triangles" complete sub-graphs with 3 points in our graph which are also sets of 3 variables, cf. Section 11.

12.5 Linear Dependencies for MQ2 Equations in DEx_2^K

Let F is the number of linearly independent equations inside our $R^{DEx} = \mathcal{O}(K^3)$ equations, it is easy to see that a fast cubic growth of R cannot translate into a curve which grows as quickly as K^3. First, the equations we generate are somewhat linearly dependent. For example it is possible to see that if we consider a complete graph (a clique) with 4 points, we will obtain 4 MQ2 equations out of which only 3 will be linearly independent. This fact alone shows that as we generate our equations, some could be omitted or they will be redundant.

In addition there is also an upper limit. It is easy to see that at no moment we can have $F > T^{DEx} = \mathcal{O}(K^2)$. It is simply impossible to have more non-redundant equations than monomials. It is important to note however that our goal is not to obtain $F/T > 1$ which is impossible. We simply want to obtain $F/T \approx 1$, cf. slide 75 in [19]. Overall we want to monitor the following quantity:

$$\frac{2K^2 - F}{2K^2}$$

and see if this quantity decreases and approaches 0, which will mean that $\lim_{K\to\infty} F/T < 1$ or if it grows slower than quadratic, this will mean that we have $\lim_{K\to\infty} F/T = 1$.

We have done computer simulations for the bitcoin elliptic curve secp256k1 and asked Microsoft Excel for the best polynomial approximation (one which would maximize the squared error). This method gives gives a remarkably good accuracy and the approximation turns out to have integer coefficients:

$$F \approx 2K^2 - 3K + 1$$

Overall we have established that:

Theorem 12.5.1. The number F of linearly independent quadratic equations and the number of terms in DEx_2^K satisfy the following approximations:

$$F \approx 2K^2 - 3K \qquad T \approx 2K^2 + K \qquad \lim_{K\to\infty} F/T = 1.$$

At this moment we can only provide experimental evidence for this result, cf. Table 6 on page below. 45.

12.6 Comparison to ElimLin

We do NOT achieve an exact equivalent of what we previously achieved with ElimLin algorithm. In Fig. 1 page 17 we have really obtained a curve which grows as K^2 This even though in the long run the quantity must become linear, because we cannot obtain more than $T = \mathcal{O}(K)$. In Section 4.3 ElimLin has 3 distinct stages: no non-trivial equations initially or $\mathcal{O}(K^0)$ and overall the number of equations found grows linearly $R = \mathcal{O}(K^1)$ due to some trivial equations. Then for a long time equations are generated at a quadratic rate $\mathcal{O}(K^2)$ which is undoubtedly faster than linear, and ultimately there are dependencies and ElimLin reaches a phase transition. It seems that the growth inevitably tends to re-become linear for large K, but this is not quite correct. The phase transition is quite fast and we only have few discrete points from which it is hard to measure the growth rate in a sensible way.

With our DEx_2^K equations the situation is a bit different, possibly for fundamental reasons outlined in Section 6.9 and Section 12.8 below. A direct comparison would be therefore misleading. Here, our F does NOT grow faster than K^2, not even temporarily, or at least we have not observed it. However, overall we achieve our objective of "saturation" all the same. Our goal in both cases is just to make F approach T asymptotically, cf. slide 75 in [19]. In both cases we achieve our primary goal, with

$$\lim_{K\to\infty} F/T = 1.$$

12.7 Next Steps?

We are able to create a system of equations which encodes the problem of EC point splitting in two. It is massively overdefined and close to saturation. The density/connectivity here is in fact clearly[22] better than in ElimLin and therefore probably better than required. However we still have a serious problem.

12.8 Limitations Due to Large Number of Solutions

Is our system of equations DEx_2^K going necessarily tend to complete saturation[23] and become solvable in polynomial time? The answer is **no**, this would be too good to be true. Unhappily, as before in Section 6.10 we cannot hope to achieve complete saturation as long our system of equations has a large number of solutions. Moreover strictly speaking the answer is yes, if we don't specify any constants all systems of equations we study are easy to solve.

Now if we code additional constraints by some extra polynomial equations, the answer will be different. We need to work on encoding constraints into such a system of equations which again is an open problem which arguably has not yet been solved in a satisfactory[24] way.

[22] This is again because each equation shares each non-linear monomial with a large number of $\mathcal{O}(K)$ of other equations.

[23] Arguably we are very close to saturation, possibly as close as one could be, given the fact that our system has plenty of solutions, we cannot really expect to do much better, cf. Section 6.9.

[24] Until now extremely very few researchers have tried to do it for mod P curves, with two notable exceptions of early work on this topic [39] and again recently in [35].

13 Computer Simulations

In this section we do NOT yet address the problem of constraints. We test our "$M = 2$ + EC Code" approach and describe additional expanded variants of it.

13.1 Computer Simulations At Degree 2

We have done a series of simulations with our equations.

The command line to run these simulations with our software for the 256-bit bitcoin elliptic curve is:

```
ec2decomp.exe 93931 256 K 1
```

This software which runs under windows command line can be found at [10]. There are two executables ec2decomp.exe and ax64.exe which need to be in the same directory. We note that the results we report in computer simulations depend on the elliptic curve **slightly** for small curves sometimes we sometimes get different results. For large curves we have never observed any difference: the results seem not depend at all on the curve choice[25].

DEx_2^K quadratic					
$2K$ value	4	8	16	32	64
$R = \#$eqs	4	48	448	3840	31744
$T = \#$mons-1	10	36	136	528	2080
F	3	21	105	465	1953
F/T	0.30	0.58	0.77	0.88	0.94
$T - F$	7	15	31	63	127
E	0	0	0	0	

Fig. 6. Simulations on DEx_2^K equations at degree 2 for the curve secp256k1

Notation. Here E is the number of Eliminators, cf. later Section 13.9.

Remark: We have

$$F = 2K^2 - 3K + 1$$

cf. Section 13.8. Moreover the number of degree 2 monomials is exactly:

$$T = 2K^2 + K$$

13.2 Faster Generation of Equations At Degree 2

An interesting question is that we have generated equations DEx_2^K in such a way that it is redundant system of equations: so far we have written $\mathcal{O}(K^3)$ equations which however have only $\mathcal{O}(K^2)$ monomials, and the rank is only at most $F \in \mathcal{O}(K^2)$ or more precisely $F = 2K^2 - 3K + 1$. An interesting question if there is an obvious way to write ONLY $\mathcal{O}(K^2)$ equations and still achieve the full rank of $F = 2K^2 - 3K + 1$. For example for $K = 2$ we could just ignore one equation out of 4 [selective erasure]. We have programmed such a simple way to achieve the maximum rank as follows:

```
ec2decomp.exe 93931 256 K 1  /writeonlyMQLimitTriangles2
```

[25] Yes MQ2 and every single equation and property we study in this paper also works for P-256, THE curve which almost everybody on this planet uses in their TLS or SSH communications, financial transactions, etc.

13.3 Completeness At Degree 2

An interesting question is whether MORE quadratic equations exist that those generated (implicit equations or relations, as apposed to the explicitly generated above by closed formulas cf. Section 11). The answer is no, we have already achieved the maximum possible rank.

implicit quadratic any set of $2K$ vars						
$2K$ value	4	8	16	32	64	128
$T = \#$mons-1	10	36	136	528	2080	8256
F	3	21	105	465	1953	
F/T	0.30	0.58	0.77	0.88	0.94	
$T - F$	7	15	31	63	127	

Fig. 7. Simulations on DEx_2^K POTENTIAL equations at degree 2.

13.4 Equations With Additional Cubic Monomials

It is possible to see that for every set of 3 variables there exist exactly one equation as in Section 11 which however also involves a product of 3 variables. We call this equation MQ3, cf. Thm. 11.1.1 in Section 11.1.

Adding this equation makes 2 equations per triangle, and in order to obtain this very basic cubic version we run:

```
ec2decomp.exe 93931 256 K 1 /sub23cubic
```

$DE^{2.3}x_2^K$ cubic			
$2K$ value	4	8	16
$R = \#$eqs	8	112	1120
$T = \#$mons-1	14	92	696
F	7	77	665
F/T	0.5	0.84	0.96
$T - F$	7	15	31

Fig. 8. Simulations on $DE^{2.3}x_2^K$ basic cubic equations for the curve secp256k1

13.5 Completeness At Degree 2.3

In our equations we allow only monomials which are products of up to three DISTINCT variables. In turns out that our equations are complete: no more equations exist with these monomials.

implicit cubic any set of $2K$			
$2K$ value	4	8	16
$T = \#$mons-1	14	92	696
F	7	77	665

Fig. 9. Simulations on $DE^3x_2^K$ POTENTIAL equations at degree 3.

13.6 Remark - No Squares

So far our equations do NOT contain squares nor multiples of squares, yet we could easily enhance our DEx_2^K equations to contain also squares and cubic terms which contain squares.

One method to this is for example by using explicit formulas of Section A.3 or those in Section A.4 in a different particular case [less general]. Such equations can also be generated by expansion of Semaev $S3$ polynomials after substitution. We have tested both methods and here we have more monomials and we do NO longer have new equations which are not consequences of the S3.

13.7 Equations At Full Degree Degree 3

This leads to a richer cubic version, or our second cubic version, in which more monomials are allowed, namely all monomials of type xi^2xj and squares of type xi^2xj. We will still not include monomials of type xi^3. In order to obtain this so called cubic version we run:

```
ec2decomp.exe 93931 256 K 1 /cubic
```

$DE^3x_2^K$ cubic				
$2K$ value	4	8	16	32
$R = \#$eqs	16	192	2240	
$T = \#$mons-1	34	156	952	
F	15	125	889	
F/T	0.50	0.80	0.93	0.98
$T - F$	15	31	63	127
E	4	56	560	4960

Fig. 10. Simulations on $DE^3x_2^K$ cubic equations for the curve secp256k1

13.8 Predicting the Outcomes

It is easy to see that for quadratic equations DEx_2^K the outcomes of all our experiments satisfy the following formula for every K:

$$T - F = 4K - 1$$

Similarly for our equations $DE^{2.3}x_2^K$ with products of 3 distinct variables allowed we have:

$$T - F = 4K - 1$$

Then for our equations $DE^3x_2^K$ expanded at degree 3 we have:

$$T - F = 8K - 1$$

These formulas give exact results 100 % of the time, no exceptions are known and no exceptions are expected. It is possible to explain these formulas by a careful analysis of linear dependencies which are quite complex here.

We have obtained at last, for all the three major forms of equations:

$$\lim_{K \to \infty} F/T = 1$$

13.9 Emerging Order and Eliminators

Definition 13.9.1 (Deficit).
We call "deficit" the $T - F$ value, in one way all monomials "live" in a linear space of dimension $T - F$.

For example we look at the cubic variant, cf. Fig. 10. These figures show an interesting phenomenon: for $K \geq 4$ we have $T - F$ which is bigger than the number of monomials of degree 0, 1 and 2 combined. We have $T - F = 31 < 28 + 8 + 1$.

This means that we can hope that ALL cubic monomials could in theory be eliminated.

In order to study this question we need the following definitions:

Definition 13.9.2 (Degree Falls).
We call degree falls equations in a linear span which have lower degree that the maximum degree in our system of equations.

Definition 13.9.3 (Eliminators).
We call eliminators a subset of our equations which contain only one monomial of maximum degree [e.g. one unique cubic monomial], and all the other monomials are of lower degree.

We sometimes denote their number by letter E.

In our DEx_2^K equations we can write formulas to eliminate cubic monomials DIRECTLY:

Theorem 13.9.4. In $DE^3x_2^K$ we can generate eliminators which allow one to **eliminate by linear algebra ALL monomials of degree 3** out of those present and replace them by equations of degree up to 2.

Proof: For every triple of variables in Thm. 11.1.1 of Section 11.1 we provide direct formula for doing this which allow after substitution of sums/differences of points to obtain Eliminators as defined above.

These basic facts are meant to convince us that we generate systems of equations which are in a state **close to saturation, with abundant degree falls**, and therefore they are [with possibly other technical conditions] very likely to be **efficiently solvable** by traditional techniques such as Gröbner bases.

Part IV

D73 EC Coding Method At Degree 3

14 Objectives for Part IV and D73 Equations

Our goal is to re-code specific forms of EC codes as systems of polynomial equations which should be as simple as possible. We start by observing that a certain simple polynomial equation exists.

14.1 D73 - A New Family of Cubic Polynomial Equations

The following result has been designed as a plausible replacement for arbitrary S3 equations in configurations with redundant "expanded" variables. We have:

Theorem 14.1.1 (D73 Theorem). We consider the following set of variables on EC, a special form of EC Code with 3 inputs and 7 outputs for any Weierstrass elliptic curve modulo a large P.

$$\begin{array}{ccc} P1 & P2 & P1+P2 \\ P1+P3 & P2+P3 & P1+P2+P3 \\ P3 & & \end{array}$$

Again we look only at x coordinates of the 7 points. We call $sx1 - sx123$ the x coordinates of the 7 points, with $sx1, sx2, sx12$ being the points the first line, with $sx13, sx23, sx123$ being the points the first line, and $sx3$ being the x coordinate for the last point $P3$. This is summarized on the picture below:

$$\begin{array}{ccc} sx1 & sx2 & sx12 \\ sx13 & sx23 & sx123 \\ sx3 & & \end{array}$$

If all the 7 points are distinct from the ECC neutral element ∞ we have:

```
sx1*sx2*(sx23-sx13) +sx1*sx3*(sx12-sx23) +sx2*sx3*(sx13-sx12)
+sx123[sx1*(sx13-sx12)+sx2*(sx12-sx23)+sx3*(sx23-sx13)] = 0
```

Remark. Our D73 equation is a homogenous polynomial of degree 3. It has very few terms. Yet another remarkable polynomial relation which we have discovered. Unlike all our previous equations such as MQ2/MQ3 and their generalizations it does NOT depend on the EC coefficients and works the same for all sorts of curves including NIST curves such as P-256. We challenge the reader to discover anything comparable in terms of elegance and simplicity for an EC Code expansion with a similar expansion factor [26] and with 3 free variables.

[26] This equation is of remarkable simplicity. A comparison to MQ2 would probably be not fair, as MQ2 has fewer active EC points. In spite of this, the new equation D73 is actually still overall shorter and simpler than MQ2 before substitution, i.e. when we look at how MQ2 depends on the dx variables. Overall the MQ2 formulas are quite complex and MQ2 has polynomials of degree up to 5 in all 6 variables, which degree would become even higher if we wanted to replace these variables by a less redundant set of 2 variables. This main point in this paper is that having redundant variables is a **good idea** and it allows to greatly simply polynomial equations and effectively replace Semaev polynomials by some simpler polynomials.

14.2 D93 - Another Remarkable Family of Homogenous Cubic Equations

In this section we introduce another class of equations which generalizes D73. We have:

Theorem 14.2.1 (D93 Theorem). We consider the following set of variables on EC, a special form of EC Code with 4 inputs and 7 outputs for any Weierstrass elliptic curve modulo a large P.

$$\begin{array}{ccc} P1-P3 & P2-P3 & \\ P1 & P2 & P1+P2 \\ P1+P3 & P2+P3 & P1+P2+P3 \\ P3 & & \end{array}$$

Again we look only at x coordinates of the 10 points. The variable numbering is as follows:

$$\begin{array}{ccc} sx11 & sx12 & \\ sx21 & sx22 & sx212 \\ sx31 & sx32 & sx312 \\ sx43 & & \end{array}$$

If all the 10 points are distinct from the ECC neutral element ∞ we have:

```
+sx12*sx22*sx212 -sx12*sx22*sx31 -sx12*sx212*sx43 +sx12*sx31*sx43
-sx11*sx21*sx212 +sx11*sx21*sx32 +sx11*sx212*sx43 -sx11*sx32*sx43
-2*sx22*sx21*sx32 +2*sx22*sx21*sx31 +sx22*sx212*sx32 -2*sx22*sx212*sx312
+sx22*sx32*sx312 -sx22*sx31*sx43 +sx22*sx312*sx43 -sx21*sx212*sx31
+2*sx21*sx212*sx312 +sx21*sx32*sx43 -sx21*sx31*sx312 -sx21*sx312*sx43
-sx212*sx32*sx312 +sx212*sx31*sx312 = 0
```

Remark. Our D93 equation is a also homogenous polynomial of degree 3. Many other classes of such cubic homogenous equations with more than 9 variables exist. We omit them due to the lack of space. Many do NOT depend on the EC coefficients and works for all the same for all sorts of curves modulo P.

f

15 The Semi-Invariant Set Method

The primary goal of Part IV is to design and built a general-purpose tool which we call the **D73 Coding**. The primary goal of D73 Coding is to take an arbitrary EC code expansion with a certain imperfect semi-invariant property and write an overdefined system of equations of degree 3 with a good R/T ratio.

With D73 Coding we do not work at degree 2 but we do work at degree 3, and the idea is again that the cryptanalyst should always remain at degree 3 maximum and avoid increasing the degree of equations which appear during the attack, this at the cost of adding additional variables. The D73 Coding is also going to be a method which more general, more powerful and more versatile than previously and can be applied in a variety of circumstances.

15.1 Expansion Objectives

One key idea is to work with arbitrary S3 equations as a starting point, maybe for those example Section 5.7 even though this might seem very difficult and quite ambitious, and expand/augmment each 3 variables by adding extra variables, by a method which we do not specify at this stage. However we DO specify the **key objective** of such expansion below.

Definition 15.1.1 (Semi-Stability of EC Codes).
We consider a family of EC Codes with a parameter K. We say that our EC Code Expansion is Semi-Stable by addition if for a pair of points selected at random in our code the probability that their sum on the EC is also present in our code is lower-bounded by a constant which does not depend on K.

15.2 Example of Semi-Stable Expansion

Here we give an example of EC Code expansion which is Semi-Stable. The reader should not think this is THE example, it is just one example which is here to illustrate our concept. On Fig. 11 below we show a simple method of expanding an arbitrary point addition into a family of EC Code expansions with a parameter K, where K is the number of lines on Fig. 11 below.

$$\begin{array}{ccc}
\vdots & \vdots & \vdots \\
P1 - 2D & P2 - 2D & P1 + P2 + 2D \\
P1 - D & P2 - D & P1 + P2 + D \\
P1 & P2 & P1 + P2 \\
P1 + D & P2 + D & P1 + P2 + D \\
P1 + 2D & P2 + 2D & P1 + P2 + 2D \\
\vdots & \vdots & \vdots
\end{array}$$

Fig. 11. Definition of RD3 Equations with Parameter K

Here if we select a point at random in the first 2 columns, with probability about 1/2 their sum is in the third column. Adding a point from the first column to a point from third column does not work, which is OK, as the probability that we want to make such an addition is constant and we are allowed to fail with a constant probability.

16 Converting Semi-Stable Expansions to Cubic Polynomial Equations Mod P

In the section we explain the general method on how the attacker can exploit the D73 type of equations. The method is a bit heuristic, or it requires a slightly stronger notion than Semi-Stability defined above. We recall that an expansion is Semi-Stable by EC addition if for a pair of points selected at random the probability that their sum is already present in our code at least a equal to a certain constant.

Definition 16.0.1 (3-Way Semi-Stability of EC Codes).
We consider any family of EC Codes with a parameter K which are semi-stable for EC point addition, cf. Def. 15.1.1 page 53.

We say that our coding method is 3-Way Semi-Stable if for a triple of points P1,P2,P3 chosen at random, the probability that the four points $P1+P2, P1+P3, P2+P3$ and $P1+P2+P3$ are simultaneously inside our initial expansion, is also lower-bounded by a fixed constant, independent on K.

Definition 16.0.2 (D73 Coding for Semi-Stable EC Codes).
For any family of EC Codes with a parameter K which are 3-Way Semi-Stable for EC point addition, cf. Def. 16.0.1 above, for each triple of points P1,P2,P3 in the initial EC Code, if the four points $P1+P2, P1+P3, P2+P3$ and $P1+P2+P3$ are simultaneously inside our EC Code expansion we write one cubic equation as specified in side Thm. 14.1.1 for each such set of 7 variables.

It is then easy to see that:

Theorem 16.0.3 (D73 Ratio Theorem). For every 3-Way Semi-Stable EC Code with $\mathcal{O}(K)$ points, our D73 Coding method produces a system with the R/T ratio which is lower bounded by a constant.

Proof: We have $\mathcal{O}(K^3)$ cases $P1, P2, P3$ and by our 3-Way Semi-Stable assumption a constant fraction produces a D73 equation. The number of cubic monomials is also $\mathcal{O}(K^3)$.

16.1 Our Compiler Tool

We have implemented a command like tool for generating these equations for an **arbitrary** EC code expansion, not only when they are 3-Way Semi-Stable[27]. It works like a compiler takes as an input ECC equations with different variables, automatically detects dependencies and outputs a set of degree 3 equations mod P. The command line is as follows:

```
ax64 4741 filename.eqs /writeD73eqs
```

The input file can look like, for example:

```
//P=0x43 A=0x0 B=0x7
P1S001+P2S001=Q12S000
P1S001+P3S003=Q13S002
P2S001+P3S003=Q23S002
P1S001+P2S001+P3S003=Q123S001
Q12S000+P3S003=Q123S001
Q13S002+P2S001=Q123S001
Q23S002+P1S001=Q123S001
```

The output file is then:

```
modulo 67
-P3S003X*P1S001X*Q12S000X +P3S003X*P1S001X*Q23S002X +P3S003X*P2S001X*Q12S000X
-P3S003X*P2S001X*Q13S002X +P3S003X*Q13S002X*Q123S001X -P3S003X*Q23S002X*Q123S001X
+P1S001X*P2S001X*Q23S002X +P1S001X*Q13S002X*Q123S001X +P2S001X*Q12S000X*Q123S001X
-P1S001X*P2S001X*Q13S002X -P2S001X*Q23S002X*Q123S001X -P1S001X*Q12S000X*Q123S001X
 = 0
```

where X signs are appended automatically to variable names.

In addition our program can also generate and append MQ2 and MQ3 equations automatically which will be combined with D73 equations:

```
ax64 4741 filename.eqs /writeD73eqs /writeTriangles7D23
```

[27] This notion does not mean anything for a fixed system of equations, it is defined only for a family of systems of equations.

16.2 A Real-Life Example of Application of Thm. 16.0.3

We have developed a full demonstrator: a method of generating a 3-Way Semi-Stable expansion and then we generate a mix of cubic equations of type D73 and MQ2/M3.

```
ec2decomp 93973 256 K  /cubic
```

$$\begin{array}{ccccccc}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
P1-D & P2-D & P3-D & P1+P2-D & P1+P3-D & P2+P3-D & P1+P2+P3-D \\
P1 & P2 & P3 & P1+P2 & P1+P3 & P2+P3 & P1+P2+P3 \\
P1+D & P2+D & P3+D & P1+P2+D & P1+P3+D & P2+P3+D & P1+P2+P3+D \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}$$

Fig. 12. An example of 3-Way Semi-Stable EC Code with K Lines

The result is exactly as predicted by Thm. 16.0.3, we get approximately:

$$\lim_{K\to\infty} F/T \approx 0.19$$

16.3 Towards an Improved Ratio

We are not quite happy with $\lim F/T \to 0.19$ obtained with D73. In the next revision of this paper we are going to study the question whether and how it is possible to achieve $\lim_{K\to\infty} F/T = 1$ by this method. We claim that this is **easy** to achieve by including also the D93 equations of Thm. 14.2 and few other disjoint families of cubic polynomial equations. In the following Section we do an initial feasibility study which could disprove our claim but cannot confirm it [this requires generating more distinct classes of equations].

16.4 Initial Feasibility Study

We start by a computer simulation which show how many MAXIMUM cubic polynomials can be generated for the family of EC Codes on Fig. 12.

It could be that there is no chance to achieve $\lim_{K\to\infty} F/T = 1$ by any method. Or that we get $\lim_{K\to\infty} F/T = 1$ and we just need to add some equations in addition to $D73$ to achieve a higher rank. We have already established that a combination of expanded MQ2/MQ3/D73 does allow to achieve a ratio F/T better than 0.2.

Here is our tool [we recommend an x64 PC with 64G RAM or more]:

```
ax64 9390419 13 -3 13377 K
```

Here K will be the number of lines on Fig. 12 above.

The answer whether or not we achieve $\lim F/T \to 1$ by this method will appear in the next update of this paper.

Part V

Discussion and Analysis

17 Conclusion

In this paper we propose new methods to expand and solve algebraic equations based on Semaev polynomials. Our primary idea is to add new variables and work on the topology of our systems of equations in order to form highly connected graphs. Our methods are very different than other approaches studied in literature. Our methods share similar objectives as many works in algebraic cryptanalysis which attempt to make the R/T ratio grow, create and exploit degree falls. However we do not expand the degree of the equations, we expand the number of variables, which is a major innovation, even though both types of methods are known to specialists in algebraic cryptanalysis, cf. Section 3.4.

Our approach is NOT to hope that some equations will maybe be efficiently solvable cf. [39, 38, 36, 35]. Our primary technique is to carefully engineer a system of equations which is likely to be efficiently solvable. We show how one can provoke a combinatorial explosion in the number of equations generated and achieve systems of equations with a high rank close to saturation. Similarly we do not just hope that some degree falls might occur which could help to solve our equations. We explicitly generate degree falls by closed formulas which allow one to efficiently generate on the fly the sort of algebraic shortcuts we may need and in the quantity we require. We obtain systems of equations which are **massively overdefined** with a very high R/T ratio which is substantially higher than in all previous works on this topic. It appears that in ALL previously published papers on this topic [39, 38, 36, 35] and many other R/T is very small. In this paper we first obtain general methods with R/T being a large constant, cf. Section 6.6 and then finally at least for simpler cases, a situation where $R/T > 1$ and F/T can approach 1 as closely as we want, cf. Thm. 12.5.1 page 43.

Our key point is that one way to produce equations which may be efficiently solvable is to to construct a family of systems of equations with

$$\lim_{K \to \infty} F/T = 1$$

We show a first proof of concept how this objective can be achieved. To the best of our knowledge this has never been accomplished before in ECDL research. We conjecture that this could be sufficient in order to solve the point splitting problem in polynomial time for $M = 2$ and possibly for larger M, required to design new algorithms to compute discrete logarithms in elliptic curves. We are far from this objective at this moment.

Current paper is essentially about methods to **transform** some polynomial equations into other larger but also more overdefined and arguably more "efficiently solvable" polynomial equations. In order achieve to the desired properties we introduced some interesting innovations: specific forms of EC codes and new types of polynomials which are NOT contained in the ideal generated by simple S3 equations which we initially used to describe the problem, cf. Thm. 10.0.3.

A **major difficulty** which we have not yet solved is how to encode contraints in such systems of equations. Currently the only attempt to solve this problem known to us is [35]. An interesting question is whether all these approaches could be combined in some advanced attacks, cf. Appendix C.

Acknowledgments: This research was inspired by numerous interesting discussions on the security of ECs I had in the last 3 years with Christophe Petit, Jean-Jacques Quisquater, Steven Galbraith, Arjen Lenstra, Antoine Joux, Louis Goubin, Jonathan Bootle and Guangyan Song.

Special Thanks to UCL Students: The following UCL students have participated in this project through study of polynomial equations in various types of elliptic curves, programming and experimentation with SAGE computer algebra software, during activities organized as a part of GA18 course on cryptanalysis taught at UCL in 2014-2016. These students are Ilyas Azeem, Iason Papapanagiotakis-Bousy, Patrick Hough Huanyu Ma, Lim Min, Wei Shao and Weixiu Tan and each of these students have done some work related to this project.

References

1. Magali Bardet, Jean-Charles Faugère and Bruno Salvy, *On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations,* in Proceedings of International Conference on Polynomial System Solving (ICPSS, Paris, France), pp. 71-75, 2004. Also known as Research report RR-5049, INRIA 2003.
2. Ludovic Perret: *Gröbner bases techniques in Cryptography,* `http://web.stevens.edu/algebraic/Files/SCPQ/SCPQ-2011-03-30-talk-Perret.pdf`
3. Jiun-Ming Chen, Nicolas Courtois and Bo-Yin Yang: *On Asymptotic Security Estimates in XL and Gröbner Bases-Related Algebraic Cryptanalysis,* In ICICS'04, LNCS 3269, pp. 401-413, Springer, 2004.
4. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*; Cryptographers' Track Rsa Conference 2001, LNCS 2020, Springer, pp. 266-281, 2001.
5. Nicolas Courtois, Gregory V. Bard: *Algebraic Cryptanalysis of the Data Encryption Standard,* In Cryptography and Coding, 11-th IMA Conference, pp. 152-169, LNCS 4887, Springer, 2007.
6. Nicolas Courtois, Adi Shamir, Jacques Patarin, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, In Advances in Cryptology, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
7. Nicolas Courtois, Jacques Patarin: *About the XL Algorithm over $GF(2)$,* Cryptographers' Track RSA 2003, San Francisco, April 13-17 2003, LNCS 2612, pp. 141-157, Springer.
8. Nicolas T. Courtois: *How Fast can be Algebraic Attacks on Block Ciphers?* In online proceedings of Dagstuhl Seminar 07021, *Symmetric Cryptography* 07-12 January 2007, E. Biham, H. Handschuh, S. Lucks, V. Rijmen (Eds.), `http://drops.dagstuhl.de/portals/index.php?semnr=07021`, ISSN 1862 - 4405, 2007. Also available from `http://eprint.iacr.org/2006/168/`.
9. Nicolas Courtois *CTC2 and Fast Algebraic Attacks on Block Ciphers Revisited* Available at `http://eprint.iacr.org/2007/152/`.
10. Nicolas Courtois: Some algebraic cryptanalysis software `http://www.cryptosystem.net/aes/tools.html`.
11. Nicolas T. Courtois, Iason Papapanagiotakis-Bousy, Pouyan Sepehrdad and Guangyan Song: *Predicting Outcomes of ElimLin Attack on Lightweight Block Cipher Simon,* In Secrypt 2016 proceedings.

12. Nicolas Courtois and Blandine Debraize: *Specific S-box Criteria in Algebraic Attacks on Block Ciphers with Several Known Plaintexts,* In WEWoRC 2007, LNCS 4945, pp 100-113, Springer, 2008.
13. Nicolas Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST,* preprint, 2010-2014, available at `http://eprint.iacr.org/2011/626`.
14. Nicolas Courtois, Pouyan Sepherdad, Petr Susil and Serge Vaudenay: *ElimLin Algorithm Revisited,* In FSE 2012, LNCS, Springer.
15. Nicolas Courtois, Jerzy A. Gawinecki, Guangyan Song: *Contradiction Immunity and Guess-Then-Determine Attacks On GOST,* In Tatra Mountains Mathematic Publications, Vol. 53 no. 3 (2012), pp. 65-79. At `http://www.sav.sk/journals/uploads/0114113604CuGaSo.pdf`.
16. Nicolas Courtois: *On Splitting a Point with Summation Polynomials in Binary Elliptic Curves,* preprint 6 January 2015, `http://eprint.iacr.org/2016/003.pdf`.
17. Nicolas T. Courtois: *New Frontier in Symmetric Cryptanalysis,* Invited talk at Indocrypt 2008, 14-17 December 2008.
    Extended version of slides presented: `http://www.nicolascourtois.com/papers/front_indocrypt08.pdf`. A much shorter version was presented at the rump session of Eurocrypt 2007, `http://www.iacr.org/conferences/eurocrypt2007/slides/rumpt26.pdf`.
18. Nicolas Courtois: *General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers,* in AES 4, LNCS 3373, pp. 67-83, Springer, 2005.
19. Nicolas Courtois: *Algebraic Attacks vs. Design of Block and Stream Ciphers,* slides used in GA18 course Cryptanalysis taught at University College London, 2014-2016, `http://www.nicolascourtois.com/papers/ algat_all_teach_2015.pdf`
20. Nicolas Courtois: *Software and Algebraic Cryptanalysis Lab, a lab used in GA18 course Cryptanalysis taught at University College London, 17 March 2016,* `http://www.nicolascourtois.com/papers/ga18/AC_Lab1_ElimLin_Simon_CTC2.pdf`
21. *Nicolas Courtois and Josef Pieprzyk: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer.
22. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Available at `http://eprint.iacr.org/2002/044/`. This preprint contains two different (earlier) versions of the XSL attack, see also [21].
23. Nicolas Courtois, Theodosis Mourouzis, Guangyan Song, Pouyan Sepehrdad and Petr Susil: *Combined Algebraic and Truncated Differential Cryptanalysis on Reduced-round Simon,* in post-proceedings of SECRYPT 2014, 28-30 August 2014, Vienna, Austria.
24. Petr Susil, Pouyan Sepehrdad, Serge Vaudenay, Nicolas Courtois: *On selection of samples in algebraic attacks and a new technique to find hidden low degree equations.* in International Journal of Information Security vol. 15 iss. 1, pp. 51-65, Springer, 2016.
25. Claus Diem: *On the discrete logarithm problem in elliptic curves,* In Compos. Math., 147(2011), pp. 75-104.
26. Jean-Charles Faugère, Ludovic Perret, Christophe Petit, and Gwenaël Renault: *Improving the complexity of index calculus algorithms in elliptic curves over binary fields,* In Eurocrypt 2012, LNCS 7237, pp. 27-44, Springer 2012.
27. Steven D. Galbraith, Pierrick Gaudry: *Recent progress on the elliptic curve discrete logarithm problem,* preprint, 22 Oct 2015, `https://eprint.iacr.org/2015/1022.pdf`

28. Darrel Hankerson, Alfred Menezes, Scott Vanstone: *Guide to Elliptic Curve Cryptography*, book, hardcover, 331 pages, Springer 2004.
29. T. Hodges, C. Petit, and J. Schlather: *First fall degree and Weil descent,* In Finite Fields Appl., 30(2014), pp. 155-177.
30. Koray Karabina: *Point Decomposition Problem in Binary Elliptic Curves,* at `https://eprint.iacr.org/2015/319`, 27 Oct 2015.
31. Ming-Deh A. Huang, Michiel Kosters, Sze Ling Yeo: *Last Fall Degree, HFE, and Weil Descent Attacks on ECDLP,* In Crypto 2015, LNCS 9215, pp. 581-600, August 2015.
32. Michiel Kosters, Sze Ling Yeo: *Notes on summation polynomials,* revised 8 Jun 2015 `http://arxiv.org/abs/1503.08001`
33. Antoine Joux, Jean-Charles Faugère: *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases*, Crypto 2003, LNCS 2729, pp. 44-60, Springer.
34. Victor Miller: *Use of elliptic curves in cryptography,* in proc. Crypto'85, pp. 417-426, LNCS 218, Springer, 1986.
35. Christophe Petit, Michiel Kosters, Ange Messeng: *Algebraic Approaches for the Elliptic Curve Discrete Logarithm Problem over Prime Fields,* In PKC 2016, vol. 2, LNCS 9615, pp. 3-18, Springer, 2016.
36. Christophe Petit and Jean-Jacques Quisquater: *On polynomial systems arising from a Weil descent,* In Asiacrypt 2012, LNCS 7658, pp. 451-466, Springer 2012.
37. Håvard Raddum: *Algebraic Analysis of the Simon Block Cipher Family,* In LatinCrypt 2015, LNCS 9230, pp. 157-169, Springer, 2015, cf. `https://www.simula.no/file/simonpaperrevisedpdf/download`.
38. Igor Semaev: *New algorithm for the discrete logarithm problem on elliptic curves,* Preprint, 10 April 2015, available at `eprint.iacr.org/2015/310/`.
39. Igor Semaev: *Summation polynomials and the discrete logarithm problem on elliptic curves,* Preprint, available at `eprint.iacr.org/2004/031/`.
40. M. Shantz and E. Teske: *Solving the elliptic curve discrete logarithm problem using Semaev polynomials, Weil descent and Gröobner basis methods - an experimental study,* In LNCS 8260, pp. 94-107, Springer 2013.

A On Generation of MQ2/MQ3 and Interesting Generalizations of MQ2/MQ3

A.1 An Example of MQ2

We give here an example of this very peculiar unique pure quadratic equation $MQ2(x1, x2, x3)$. This example was computed for the bitcoin elliptic curve secp256k1 and can be seen a computing a unique equation true with probability 1 each time 3 difference assumptions on the 3 variables are simultaneously true.

```
(ix1,.)-(ix2,.)=(151714334927741905037485499455834360578447566596481491651337570427948243750 60,
67640625960982344923913992057504513842322505602249806174879684644326606374127)
(ix1,.)+(ix3,.)=(811492673006401408865805028932284068353577361537865775822378616399493255049 41,
75095704628746187771964810120718853818098264899310447405074798889107204806 30)
(ix2,.)+(ix3,.)=(142467183728907549694479635684153334994509942320601438883513939935941497984 40,
11396640204581428302983725168794405186533530038054115601131802539181159932818 4)

435066035945094763948405946891735340634024670038502487051730106844758607074 25 +
-158506579580195313723857262068843540876381538481962986023981010111697905549 98*(ix1) +
-390914182621567871576414803336485778944069050785352372697494189000092579591 01*(ix2) +
-525267644164425415364017301006644319213323723378134910935147009834883219277 7*(ix3) +
-220264235212323977692286959461405903368371413044987242998202603608515094689 40*(ix1*ix2) +
-445385506029413084626698432727559096646283103238424864906679462611919104960 60*(ix1*ix3) +
 1*(ix2*ix3) = 0
```

We also note that there is no terms of type v^2 in this equation even though such terms are quadratic. It has only 7 monomials.

A.2 Closed Formulas For MQ2 and MQ3 Equations

The main result and closed formulas appear in the main paper, cf. Section 11 and Section 11.1.

A.3 MQ3^2 Equations with Squares

Similar to MQ2 and MQ3 equations, we also have:

```
A=0, B=7 =>
-308*(dx12*dx13+dx12*dx23+dx13*dx23)  +
(ix1+ix2+ix3)*(-224*dx12-224*dx13-224*dx23)  +
ix1^2*(3*dx12*dx13^2+3*dx12^2*dx13-dx12^2*dx23-dx13^2*dx23+2*dx12*dx13*dx23)  +
ix2^2*(3*dx12*dx23^2-dx13*dx23^2-dx12^2*dx13+3*dx12^2*dx23+2*dx12*dx13*dx23)  +
ix3^2*(-dx12*dx13^2-dx12*dx23^2+3*dx13*dx23^2+3*dx13^2*dx23+2*dx12*dx13*dx23)  +
ix1*ix2*(10*dx12*dx13^2+10*dx12*dx23^2-10*dx13*dx23^2-2*dx12^2*dx13
        -2*dx12^2*dx23-10*dx13^2*dx23-2*dx12*dx13*dx23)  +
ix1*ix3*(-2*dx12*dx13^2-10*dx12*dx23^2+10*dx13*dx23^2+10*dx12^2*dx13
        -10*dx12^2*dx23-2*dx13^2*dx23-2*dx12*dx13*dx23)  +
ix2*ix3*(-10*dx12*dx13^2-2*dx12*dx23^2-2*dx13*dx23^2-10*dx12^2*dx13
        +10*dx12^2*dx23+10*dx13^2*dx23-2*dx12*dx13*dx23)  +
ix1*(ix2^2+ix3^2)*(-2*dx23^2+dx12*dx13-dx12*dx23-dx13*dx23)  +
ix2*(ix3^2+ix1^2)*(-2*dx13^2-dx12*dx13+dx12*dx23-dx13*dx23)  +
ix3*(ix1^2+ix2^2)*(-2*dx12^2-dx12*dx13-dx12*dx23+dx13*dx23)  = 0
```

There are several such equations, some with higher degree/complexity in the dx, and which are possibly redundant. We show one which is unusually simple and elegant.

One special case of these equations can be found in Section A.4. If we substitute the $dx1$, we obtain typically 7 linearly independent equations which number allows one to find special cases of equations which eliminate many monomials of high degree. For example out of 7 we can generate MQ2, one MQ3, and three will be examples for Thm. C.0.4. All these cases uses substantially less cubic monomials [and MQ2 uses none].

A.4 More Specific Equations With Squares - A Simple Case With 3 Points

Here is a special case of a triangle, where more than the usual equations MQ2 and MQ3 exist, and there exist an additional equation with squares which is is somewhat unusually simple and contains very few monomials.

$$P1 - D$$
$$P1$$
$$P1 + D$$

Fig. 13. One Possible Regular Expansion of 1 Variable

We get:

```
+4B
+dx1^2*(2*ix2-ix1-ix3)
+ix2^2*(2*dx1-ix1-ix3)
+2*ix2*(ix1+ix3)*dx1 = 0
```

Remark 1. Here there is a symmetry which makes that x1 and x3 always appear together with same coefficients.

Remark 2. If we substitute the $dx1$, the existence of such an equation is due to Cor. C.0.4.

B Higher Degree Multivariate Eliminators and Generalizations of MQ2 and MQ3

We consider equations which contain only monomials of degrees $0, 1$ and d for every $d+1$ points. We call these equations A-d-H equations (mix of Affine and degree d Homogenous terms). They are a generalization of MQ2.

Theorem B.0.1. As previously such equations are unique and exist for every d.

Similarly, we also have another set of unique equations which contain monomials of degrees $0, 1$ and d and $d+1$, which generalize MQ3 equations for every $d+1$ points. They can also be used to ELIMINATE any monomial of ANY degree with an arbitrary number of variables and replace it by relatively small number of lower degree and linear monomials.

We give here one example of such equation we generated with $d = 4$.

```
-2911649*(1)+985083*(ix1)-63765*(ix2)-3621022*(ix3)-2932833*(ix4)-3392556*(ix5)
+2091767*(ix1*ix2*ix3*ix4)+1590098*(ix1*ix2*ix3*ix5)+303909*(ix1*ix2*ix4*ix5)
-2441608*(ix1*ix3*ix4*ix5)+1*(ix2*ix3*ix4*ix5) = 0
```

C Equations with High Powers in One Variable

This section is meant to demonstrate the power of our technique which consists of adding new variables instead of expanding the degree of equations. The primary goal below is to show that EC codes do a job in some remote sense polynomial-algebraically equivalent to increasing the degree of polynomials with one[28] variable motivated by the idea that we need such high powers to encode constraints, cf. [35].

For simplicity let $x1 = Z_{10}$ and $x2 = Z_{20}$ and $x3$ by ANY other variable different than $x1, x2$. We have:

Theorem C.0.2. For every N if $2K \geq 2N-1$ there exists an equation of type:

$$x1^N \text{AffineSum}(2K-1\text{variables excluding } \ x1) = \sum (DEx_2^K \ \text{ monomials})$$

Where $\sum (DEx_2^K monomials)$ is a quadratic equation which belongs to DEx_2^K. discovered by theory, confirmed by simulations

This equations becomes quadratic if we consider that $x1^N$ is a new variable.

Moreover, with 2 more variables, it is possible to avoid x1 to appear inside the quadratic part. The variable x1 will then appear only once.

For example:

Corollary C.0.3. If $2K \geq 7$ there exists an equation of type:

$$x1^4(1 + ax2 + bx3 + cx4 + dx5 + dx6) = \sum (DEx_2^K \ \text{ monomials})$$

Moreover, with $2K \geq 9$ variables, it is possible to avoid x1 inside the quadratic part.

Corollary C.0.4. If $2K \geq 3$ there exists an equation of type:

$$x1^2(1 + ax2 + bx3) = \sum (DEx_2^K \ \text{ monomials})$$

Moreover, with $2K \geq 5$ variables, it is possible to avoid x1 inside the quadratic part.

C.1 Eliminating $x1^1 = x1$

The simplest case is as follows:

Corollary C.1.1. If $2K \geq 1$ there exists an equation of type:

$$x1(1 + ax2 + bx3) = \sum (DEx_2^K \ \text{ monomials})$$

Moreover, with $2K \geq 3$ variables, it is possible to avoid x1 inside the quadratic part and we get

$$x1(1 + ax2 + bx3) = cx2x3 + dx2 + ex3 + f$$

The last equation is simply just what we called MQ2 elsewhere and the explicit formulas to write this equation are given in Section 11.

[28] The reader should also read Section 3.5 which shows the same thing for multivariate polynomials.